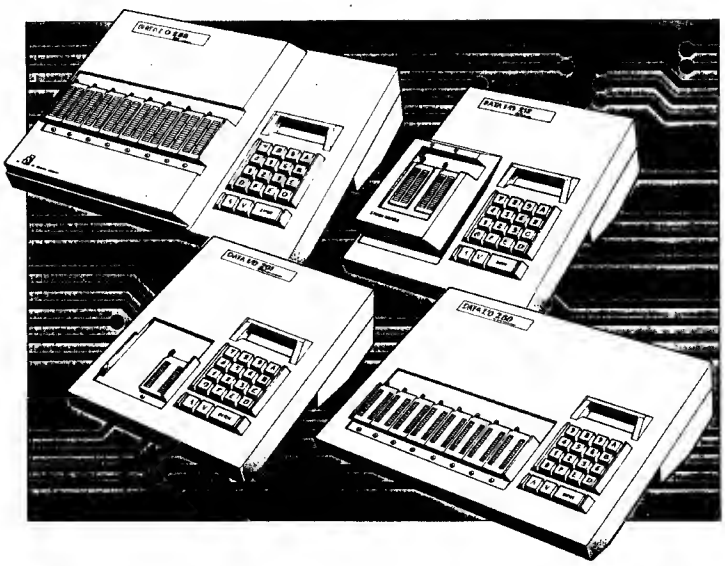
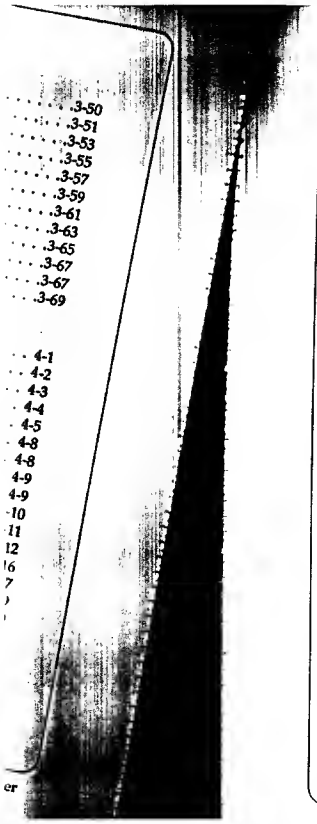
**DATA I/O** 200 Series



**DATA I/O**

# TABLE OF CONTENTS



## SPECIFICATIONS

Specifications for the 288A Multi Programmer are listed below.

## Functional Specifications

Functional specifications for the 288A are as follows:

Data RAM: 512K or 2M (bytes)

| | |
|---|---|
| Translation Formats: | ASCII Space Hex<br>Binary<br>Intel 32-bit Hexadecimal<br>Intel Intellec 8/MDS<br>Intel MCS-86 Hexadecimal Object<br>Motorola Exorciser (S1)<br>Motorola Exormax (S1 and S2)<br>Motorola 32-bit (S1, S2, and S3)<br>Tektronix Hexadecimal |

Input/Output: Serial RS-232C compatible (110 through 19.2K baud)

| | |
|---|---|
| Remote Control: | Computer Remote Control (CRC)<br>Terminal Remote Control (TRC) |

Device Sockets: Eight

Keyboard: 16-key hexadecimal, 3-key control

Display: 16 x 2 character alphanumeric LCD display with adjustable display angle

# INTRODUCTION

The 288A Multi Programmer is a gang programmer which accepts socket modules that allow you to program up to eight identical devices, or a set of up to eight devices containing different data. The 288A Multi Programmer allows you to program different types of devices simply by installing a different socket module. For example, with the 32-pin module installed you can program 24-pin, 28-pin, and 32-pin NMOS and CMOS EPROMs and EEPROMs. With the 40-pin module installed you can program 40-pin megabit EPROMs.

You can operate the 288A Multi Programmer "locally," using the front panel keys and 32-character display, or "remotely" using a terminal or computer and the RS-232C serial I/O port. In addition to the basic programming features, the 288A also includes features which allow you to edit data loaded into the programmer's RAM, make use of electronic IDs, blank check devices, and work with data in 8-bit wide, 16-bit wide or 32-bit wide words.

The 288A offers nine data translation formats (such as ASCII Space Hex, Binary, and formats supplied by Intel, Motorola and Tektronix) which enable you to transfer files to and accept files from software development systems.

This manual contains the instructions necessary for operating the 288A Multi Programmer both locally (from the front panel) and remotely. Instructions for setting up the programmer for either remote or local operation are provided in the Getting Started section and operating instructions for each of the modes are provided in separate, tabbed sections.

*NOTE*

*Additional documentation may have been included with your module. This supplementary documentation contains instructions specific to the module and should be read before performing any operations with the module.*

## ORDERING OPTIONS

The following paragraphs describe the available 288A Multi Programmer systems, options and feature updates. If you are interested in purchasing another complete system, one or more of the options, or an update, contact your nearest Data I/O sales representative. A list of representatives is included in this manual behind the index. An order for shipment must include the following information:

- Description of the equipment
- Quantity of each item ordered
- Shipping and billing address of firm, including ZIP code
- Name of person ordering equipment
- Purchase order number
- Desired method of shipment

## Power Requirements

Power requirements for the 288A are as follows:

- Operating Voltages: 90 V — 130 V or 180 V — 260 V
- Frequency Range: 48 — 62 Hz
- Power Consumption: 165 W nominal

## Physical and Environmental

Physical and environmental requirements for the 288A are as follows:

- Dimensions: 31 cm x 11 cm x 36 cm (12.2" x 4.3" x 14.2")
- Weight: Base — 2.9 kg (6.4 lbs); Module — 0.9 kg (2.0 lbs)
- Operating Temperature Range: 5 to 45 C (41 to 113 F)
- Storage Temperature Range: -40 to 70 C (-4 to 158 F)
- Humidity: Up to 90%, noncondensing
- Operational Altitude: To 10,000 ft

## System and Hardware Options

The following table lists the available 288A Multi Programmer systems and the socket modules which can be installed on the 288A base unit. Use the model numbers listed below when ordering an additional system or option. To increase the number or types of devices that your 288A Multi Programmer can program, you can install an alternate socket module on the programmer. The available socket modules are listed below.

| Model Number | Description |
|---|---|
| 288L1 | 288A Multi Programmer base unit with 512K of data RAM, configured for low voltage supplies (90 V — 130 V). Does not include a socket module. |
| 288H1 | 288A Multi Programmer base unit with 512K of data RAM, configured for high voltage supplies (180 V — 260 V). Does not include a socket module. |
| 288L2 | 288A Multi Programmer base unit with 2 Mbytes of data RAM, configured for low voltage supplies (90 V — 130 V). Does not include a socket module. |
| 288H2 | 288A Multi Programmer base unit with 2 Mbytes of data RAM, configured for high voltage supplies (180 V — 260 V). Does not include a socket module. |
| MOD32 | 32-pin MOS Module — allows duplication and set programming of 24-pin, 28-pin, and 32-pin EPROMs and EEPROMs. |
| MOD40 | 40-pin Megabit Module — allows duplication and set programming of 40-pin megabit EPROMs. |
| MOD286870X | 28-pin Micro Module — allows duplication of programmable Motorola 28-pin microprocessors. |
| MOD406870X | 40-pin Micro Module — allows duplication of programmable Motorola 40-pin microprocessors. |

| | |
|---|---|
| MOD87XX | 40-pin Micro Module — allows duplication of programmable Intel 40-pin microcontrollers. |

Other socket modules or options may become available in future. Contact your local Data I/O sales representative for updated information on available 288A options.

## Programming Capability Updates

As new device programming capabilities become available, Data I/O will offer user-installable "memory card update kits" for the 288A Multi Programmers. These update kits will include a memory card (which includes all of the updated 288A software) for all of the 288A models. Contact your local Data I/O sales representative for information on available updates. Data I/O also offers Full Service Agreements which provide automatic shipment of updates, plus calibration and repair of your equipment. Contact your local Data I/O representative for information on service contracts.

## Data I/O Device Support Policy/Liability

1. Data I/O strives to achieve more device support approvals from semiconductor manufacturers than any other programmer manufacturer.

2. Every effort is made to program an adequate number of samples according to the manufacturer supplied specification, and verify waveforms as per that specification prior to release of support. Manufacturers' approvals are to be sought in parallel with this process.

3. Data I/O's objective is to seek and obtain approvals on all devices.

4. Data I/O has made every attempt to ensure that the device information (as provided by the device manufacturer) contained in our programmers, software and documentation is accurate and complete. However, Data I/O assumes no liability for errors, or for any damages, whether direct, indirect, consequential or incidental, that result from use of documents provided with equipment or from the equipment or software which it accompanies, regardless of whether or not Data I/O has been advised of the possibility of such loss or damage.

## Customer Support BBS

In addition to calling the Data I/O Customer Resource Center, you can also call Data I/O's Customer Support Electronic Bulletin Board System.

From the Customer Support BBS you can obtain a wide range of information on Data I/O products, including current product information, new revision information, known bugs (and work-arounds), helpful application notes, and other miscellaneous information. In addition, the BBS has a collection of DOS utilities you can download.

The Customer Support BBS also has a message facility which allows you to leave messages to Customer Support Personnel. For example, you could request support for a specific device, or suggest how we can improve our products. Or you could leave a message telling us what you think of Data I/O product(s).

To learn more about the Data I/O Customer Support BBS, call it at (206) 882-3211. The protocol is 1200/2400/9600 (Courier HST) baud, 8 data bits, 1 stop bit, and no parity. On-line help files are available throughout the BBS to help you learn more about the BBS.

## Warranty Information

### Description

Data I/O warrants its products against defects in materials and workmanship for a period of one (1) year for hardware and ninety (90) days for software unless specified otherwise, which begins when the equipment is shipped. Refer to the warranty card inside the back cover of this manual for information on the length and conditions of the warranty.

# CUSTOMER SERVICE

## Contacting Customer Support

In the United States, the Data I/O Customer Resource Center is staffed with Support Engineers between 6:00 AM and 5:00 PM Pacific Time. Outside the United States, you should call your local Data I/O representative.

Regardless of whether you call Data I/O or your local Data I/O representative, you can ensure quick and accurate phone assistance by following the steps below:

1. Have your programmer and/or computer in front of you.
2. Have this manual available.
3. Be ready to provide the following information:
   - Manufacturer and part number of the device you are using.
   - Error codes and messages exactly as they appeared.
   - Detailed explanation of the problem you are experiencing.
   - Version number of your Data I/O product; to find this number, powerup the programmer. See the Operator's Manual Section, *Getting Started*.

In the United States, the phone number for the Data I/O Customer Resource Center is (800) 247-5700. Outside the United States, you should refer to the Warranty Service section of this chapter for the phone number of your local Data I/O representative.

# MANUAL CONTENTS

A brief outline of the contents of this manual is provided below:

- INTRODUCTION—This section provides a general description of the programmer, its specifications, available options and ordering information, and warranty and service information.
- GETTING STARTED—This section provides instructions on how to set up the 288A Multi Programmer for operation and a sample programming session to give you a general idea of how to operate the programmer.
- FRONT PANEL OPERATION—This section provides detailed instructions on how to blank check, program and verify devices using the front panel keys. How to perform operations on single devices, "gangs" of identical devices, and sets of devices using the front panel keys is explained, as well as how to upload data, prepare the programmer to accept and store downloaded data, and how to edit data stored in the programmer's RAM using the front panel keys.
- TERMINAL REMOTE CONTROL—This section provides instructions on how to operate the 288A Multi Programmer from a remote terminal connected to the programmer through the RS-232C port.
- COMPUTER REMOTE CONTROL—This section provides a description of the computer remote control command language. This command language can be used to write a software driver that allows you to operate the programmer using a host computer. This section also contains detailed descriptions of the data translation formats available for the 288A Multi Programmer.
- ERROR MESSAGES/DEVICE LIST—This section provides a list of the error messages displayed on the programmer. The meanings of the error messages are described as well as corrective action that should be taken. The device list, provided separately, should be placed in this section as well.
- INDEX—This is an alphabetical guide to all major topics covered in this manual.

## Warranty Service

Data I/O maintains customer support centers throughout the world, each staffed with factory-trained technicians to provide prompt, quality service. This includes not only repairs, but calibration checks of all Data I/O products. For warranty service, contact your nearest Data I/O Customer Support Center. If you return the unit for service, please return the disks with the unit. The following is a list of Data I/O offices:

**United States**
Data I/O Corporation
10525 Willows Road N.E.
P.O. Box 97046
Redmond, WA 98073-9746
Telephone: (206) 881-6444
(800) 426-1045
Fax: (206) 882-1043
Telex: 152167 (Inside U.S.)
4740166 (Outside U.S.)

**U.S. Customer Resource Center**
(800) 247-5700

**Customer Support BBS**
(206) 882-3211

**Data I/O New Hampshire**
20 Cotton Road
Nashua, NH 03063
Telephone (603) 889-8511
(800) 858-5803 (NJ & NY only)
Fax: (603) 880-0697

**Data I/O San Jose**
1701 Fox Drive
San Jose, CA 95131
Telephone (408) 437-9600
Fax: (408) 437-1218

**Data I/O Japan**
Sumitomoseimei
Higashishinbashi Bldg. 8F
2-1-7, Higashi-Shinbashi
Minato-Ku, Tokyo 105, Japan
Telephone: (03) 432-6991
Fax: (03) 432-6094 (Sales)
(03) 432-6093 (Other)
Telex: 2522685 DATAIO J

**Data I/O Canada**
6725 Airport Road, Suite 302
Mississauga, Ontario
L4V 1V2 Canada
Telephone: (416) 678-0761
Fax: (416) 678-7306

**Data I/O Europe**
World Trade Center
Strawinskylaan 633
1077 XX Amsterdam
Telephone: +31 (0)20-6622866
Fax: +31 (0)20-6624427
Telex: 16616 DATIO NL

**Data I/O-Instrumatic Electronic Systems Vertriebs GmbH**
Lochhammer Schlag 5a
D-8032 Gräelfing
West Germany
Telephone (0)89 858580
Fax: (0)89 8585810

## Socket Warranty

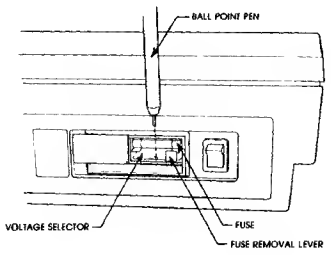
Sockets are warranted for 25,000 cycles. Socket replacements sre available through the Data I/O Service Center or local Data I/O Representative.

To install the line voltage selector (or change the line voltage) and check (or change) the line fuse, proceed as follows:

*CAUTION*

***This instrument may be damaged if operated with the wrong line voltage.***

1. Disconnect the power cord if plugged into the unit. If the power cord has never been plugged into the unit, the power receptacle will be covered by a cautionary label. Remove the label.
2. Slide open the clear plastic door that covers the fuse.
3. Remove the fuse by pulling on the lever labelled FUSE PULL (see illustration).
4. If already installed, remove the voltage selector (small printed circuit board) by prying it out of its slot with a sturdy pointed object (see illustration). If the voltage selector has never been installed, remove it from the voltage kit bag.



# 1. GETTING STARTED

This section explains how to install the line voltage selector included with your 288A (or change the operating voltage and line fuse), how to ground the unit, how to connect the RS-232C port for remote operation, how to install a socket module, and how to power up the unit. Also given in this section are instructions for obtaining the firmware configuration number, a brief description of the 288A menus and a sample EPROM programming session.

## CONNECTING POWER

Before connecting power to the programmer, perform the following checks:

- ☐ Make sure the operating voltage is properly selected on the back panel of the unit.
- ☐ Make sure the correct line fuse is installed.
- ☐ Make sure the unit is properly grounded.
- ☐ Make sure the memory card is fully inserted in the slot on the right-hand side of the unit.

### Installing/Changing the Line Voltage Selector and Fuse

The line voltage on which the 288A is to be operated is selected by installing a voltage selector card in the back of the programmer. The voltage selector card is enclosed with your 288A base unit in the voltage kit. The voltage kit also includes a spare line fuse which can be used if the installed fuse is blown. The line voltage selector is a small printed circuit board that can be inserted into its slot in any of four different positions to accommodate 100, 120, 220, and 240 line voltages.

## Grounding the Programmer

The 288A is shipped with a three-wire power cable. This cable connects the chassis of the programmer to earth ground when connected to a properly grounded three-wire ac receptacle.

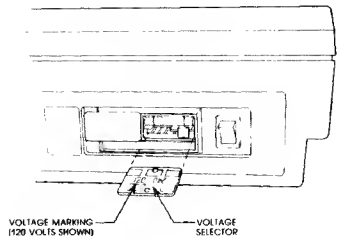
WARNING

**Continuity of the grounding circuit is vital for the safe operation of the unit. Never operate this equipment with the grounding conductor disconnected.**

5. Orient the voltage selector so that the desired voltage marking is on the top surface of the selector and to the left side, as shown in the illustration.

6. Insert the voltage selector into its slot and push it firmly into place. If the selector is not inserted fully into its slot, no power connection will be made.

7. Verify that the fuse is the correct type and value per the following table then install the correct fuse into the fuse holder.




| | Line Fuse Rating | | | |
|---|---|---|---|---|
| Line Voltage | Current | Voltage | Type | Data I/O Part Number |
| 90V — 260V | 1.5A | 250V | Slo-blow* | 416-3012-001 |

* Littlefuse type 313, Bussman type MDX

*NOTE*

*The line fuses included with your programmer are 1/4 x 1-1/4 inch fuses. The fuse holder can also accept 5 x 20 millimeter fuses, commonly available in Europe.*

8. Slide the door closed.

*CAUTION*

***For continued protection against the possibility of fire, replace only with a fuse of specified current, voltage, and type ratings.***

# INSTALLING AND REMOVING SOCKET MODULES

The socket modules contain the sockets in which the devices to be duplicated are installed. Different socket modules allow you to duplicate different kinds of devices. The programmer cannot be operated without a socket module installed. The socket module can be removed and another module installed while power is on, but it is recommended that you install a socket module prior to powering up the programmer. To install and remove the socket module, perform the following procedures.

## Installing a Socket Module

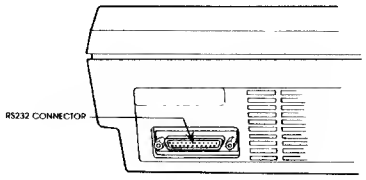
1. Insert the retaining hook on the underside of the socket module into the hole at the top of the 288A front panel (see illustration).

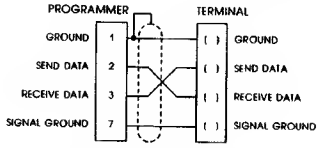
2. Lower the front of the module slightly, slide the module towards the back of the base unit as far as the retaining hook will allow and then continue lowering the front of the module until the module drops into place. The module should lower down until the bottom of the module is resting almost fully against the top of the 288A base unit.

3. Press firmly on the lower front portion of the module to seat the connector on the bottom of the module into the 288A's connector (see illustration). If the connector is not fully engaged, the programmer will not operate and a "MODULE DISCONNECTED" message will appear on the front panel display.

# RS-232C PORT CABLE CONNECTIONS

The 288A is equipped with one serial RS-232C compatible I/O port which is located on the back panel of the programmer (see illustration). The RS-232C port can be linked to a terminal, computer or other development system in either a hardware handshake or non-handshake mode. The cable connections required to the RS-232C connector for each mode are shown in the following illustration.

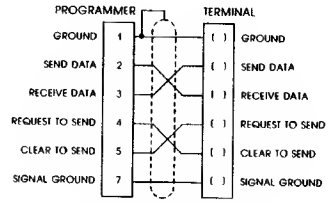



Half/Full Duplex, Without Handshake

*NOTES*

1. *All signals are named with respect to the originating unit*
2. *All undesignated pins are to be left open*
3. *For applications that do not require handshaking the programmer's clear to send line is pulled up internally*
4. *Host system's pin numbers may differ*




Half/Full Duplex, With Handshake

When the programmer is turned on, it automatically performs a self test that verifies correct operation of the unit. During execution of the self test, the front panel display shows

```
SELF TESTING
```

with advancing decimal points on the lower portion of the display.

If a module is not installed when the power is switched on, the display reads:

```
  MODULE
DISCONNECTED
```

Install a module. The self test will then proceed.

If a device is present in any of the programmer's sockets when the power is switched on, the display reads

```
DEVICE IN SOCKET
 REMOVE DEVICE
```

To correct this condition, remove the device and press ENTER. The self test will then proceed.

Upon completion of the self test, the display shows

```
  SELF TEST OK
DATA I/O 288A n
```

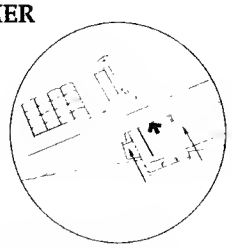
where "n" is the version number of the programmer's software. The programmer then displays the first main menu item:

```
LOAD FROM MASTER.
```

## POWERING UP THE PROGRAMMER

To power up the programmer, proceed as follows:

1. Make sure the device sockets are empty.
2. Make sure the memory card is fully inserted in the programmer (see illustration).



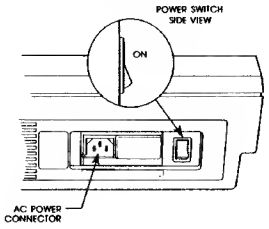
*NOTE*

*Applying power without the memory card inserted will either cause no response from the programmer or will cause the programmer to beep and display message: INSERT MEMORY CARD*

*CAUTION*

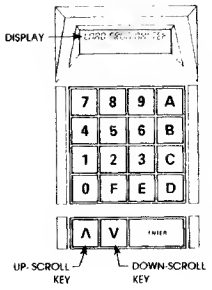
***Removing or installing memory card with power applied may result in damage to the memory card.***

3. Plug the power cable into the back panel connector and an AC power receptacle.
4. Press the power switch to the ON position (see illustration).



## THE MAIN MENU FUNCTIONS

The programmer can perform seven basic functions, each of which is presented on the main menu. To display each of these functions, scroll through the main menu by means of the scroll keys shown in the illustration. (Scroll forward with the right-hand key and backward with the left-hand key.) The seven main menu functions are:




LOAD FROM MASTER — load data from a master device into internal memory.

LOAD FROM SET— load data from a set of master devices into internal memory.

PROGRAM — program one or more devices, or sets of devices with the contents of internal memory.

VERIFY— verify the contents of one or more devices against the contents of internal memory.

BLANK CHECK — perform a blank check on the installed device(s).

RS232 — perform RS-232C port operations, such as change the port settings, download a file, upload a file, or transfer control of the programmer to a remote computer or terminal.

EDIT — edit the contents of the programmer's data RAM.

Each of the seven functions is fully described in the Front Panel Operation section; however, to familiarize you with the basic programming operation, a sample programming session is described in the following pages.

## Displaying the Firmware Configuration Number

You will need to know the firmware configuration number of your 288A when contacting Data I/O service personnel. To cause the programmer to display its firmware configuration number during the power up sequence, hold down the up-scroll and ENTER keys when you press the power switch to the ON position. The programmer displays

```
SELF TESTING
```

with advancing decimal points on the lower portion of the display. When the programmer displays

```
LCD DISPLAY TEST
```

press any front panel key twice. The programmer will then display

```
    FIRMWARE
SUMCHECK = HHHHHH
```

where "HHHHHH" is the firmware sumcheck, or configuration number, of the programmer's firmware. To begin normal programmer operation, turn the programmer power off and then on again.

5. When the correct manufacturer is displayed, press ENTER to select that manufacturer. If you selected Intel as the manufacturer, the display would read

```
LOAD FROM MASTER
  INTEL  2716
```

*NOTE*

*If you are using a device of a manufacturer other than Intel, the name of that manufacturer and a device part number appear on the display.*

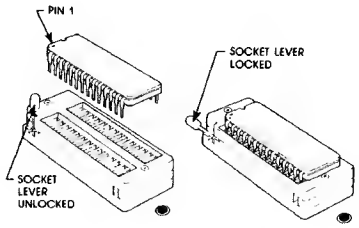
6. Press the up-scroll or down-scroll key until the display shows the correct part number of your master device. For example, if you are using an Intel 2764, the display should read

```
LOAD FROM MASTER
   INTEL 2764
```

7. Press ENTER to select the device type. The display reads

```
INSERT DEVICE
 INTEL 2764
```

8. Insert the master device in the leftmost socket (socket 1) by lifting the lever and placing the device in the socket so that the bottom pins of the part are at the bottom of the socket and pin 1 is toward the top of the socket (see illustration). Push the lever down to lock the device in place.



## SAMPLE PROGRAMMING SESSION

The following steps describe how to program a blank device from a master device. (A master device is a device that has been previously programmed and is used as a "master" to program blank devices). To perform this programming session, you will need a master device and a blank device. The master device used in the following procedure is an Intel 2764, but could be any device shown on the Device List.

In the following procedure, the blank device is assumed to be of the same type as the master device, although it is possible for the master device to differ from the blank device(s). For more details on device programming, refer to the Front Panel Operation section of this manual.

1. If the programmer is not on, make sure all the sockets of the programmer are empty and turn on the power switch. Wait until the self test is complete and the display reads

   ```
   LOAD FROM MASTER
   ```

2. Press ENTER to select the LOAD FROM MASTER function. The display reads

   ```
   LOAD FROM MASTER
    F/P CODE FF/FF
   ```

   "FF/FF" is the current family/pinout code. (A family/pinout code of FF/FF causes the programmer to read the electronic ID of the installed device and select the correct family/pinout code automatically.) For this sample session, select the device type from the 288A menus.

3. Press the down-scroll key once. The display reads

   ```
   LOAD FROM MASTER
     ELECTRONIC ID
   ```

4. Press the down-scroll key until the display shows the name of the manufacturer of the master device you are loading.

   *NOTE*

   *If you scroll past the desired selection, use the up-scroll key to move backwards through the menu.*

14. Press the ENTER key if the displayed set size is 1; or, if 1 is not displayed, press the scroll keys until a set size of 1 is displayed and press the ENTER key. The display reads

```
INSERT   DEVICE
F/P CODE 79/33
```

15. Press the ENTER key to initiate the programming operation. During the programming operation the display reads

```
      PROGRAM
................
```

(Advancing decimal points on the lower portion of the display indicate that the program operation is taking place.)

When the program operation is complete the display reads

```
      PROGRAM
SUMCHECK = HHHHHH
```

where "HHHHHH" is the sumcheck of the device. This hexadecimal number should match that displayed in step 9.

16. Lift up the socket lever and remove the programmed device.

This completes the sample programming session.

9. Press ENTER to begin the load operation. The display reads

```
LOAD FROM MASTER
.............
```

(Advancing decimal points on the lower portion of the display indicate that the load operation is taking place.)

When the load operation is complete the display reads

```
LOAD FROM MASTER
SUMCHECK = HHHHHH
```

where "HHHHHH" is the sumcheck of the device (the sum of all the data bytes in the device expressed as a hexadecimal number) or devices (for set operations). Make a note of the sumcheck so that the device programmed later in this session can be verified.

10. Lift the socket lever and replace the master device with a blank device. Push the socket lever down.

11. Press the down-scroll key three times. The display reads

```
PROGRAM
```

12. Press the ENTER key to select the program function. The display reads

```
PROGRAM
F/P CODE 79/33
```

(79/33 is the family/pinout code for the Intel 2764, and is now the default family/pinout code. If you used a master device other than an Intel 2764, the family/pinout code of the device you selected is now the default.)

13. Press the ENTER key to select the displayed family/pinout code. The display reads

```
PROGRAM
SET SIZE = 1
```

# GENERAL OPERATING NOTES

The following notes explain features, functions, and displays that are common to nearly all of the front panel operations.

## Family/Pinout Codes and Device Part Numbers

The required programming algorithm and the pin assignments of devices differ greatly. In order for the programmer to program a specific device, the device to be programmed must be selected on the programmer. Device selection may be made by selecting the manufacturer and part number from the 288A menus, or by entering a specific family/pinout code on the front panel keypad. The operating procedures presented in this section allow you to select the device by either method.

*NOTE*

*The device's electronic identifier may also be used to identify the device to the programmer (see the subsection "Devices with Electronic Identifiers").*

To allow for additional family/pinout codes in the future, Data I/O has gone to a six-digit family/pinout code on the front panel menu. Four- or six-digit codes are accepted using the following method.

| Procedure | Example 288A Displays |
|---|---|
| 1. To enter family/pinout code AF/33, press A, F, ENTER, then press 3, 3, ENTER | |
| The display reads | INSERT DEVICE<br>AF/33 |
| or | |

# 2. FRONT PANEL OPERATION

This section describes how to blank check, program and verify parts, and sets of parts, using the front panel keys and display of the 288A Multi Programmer. Also provided are instructions for using the RS-232C serial interface port in conjunction with the front panel keys; i.e., setting the RS-232C communications protocol and performing download and upload operations. The information in this section is divided into the following subsections:

- GENERAL OPERATING NOTES — Provides general information pertaining to performance of many of the 288A's operations and functions. Read this section before attempting any 288A operations.
- CHECKING FOR NON-BLANK DEVICES — Describes how to verify that a device is completely blank (contains no data).
- PROGRAMMING FROM A SINGLE MASTER — Describes load and program operations required to individually program and gang program devices.
- PROGRAMMING FROM A SET OF MASTER DEVICES — Describes load and program nperations required to program sets of devices.
- VERIFYING PROGRAMMED PARTS — Describes how to verify the data programmed into a device against the contents of the programmer's memory.
- SETTING COMMUNICATIONS PROTOCOL — Describes how to set the RS-232C communications protocol, such as the baud rate and the number of stop bits, for the 288A.
- DOWNLOADING DATA — Describes how to prepare the programmer to receive and store data transferred from a remote data source (computer or development system) to the programmer's RAM.
- UPLOADING DATA — Describes how to perform the transfer of data from the programmer's memory to a host computer or other development system.
- EDITING RAM DATA — Describes how to edit the contents of the programmer's data memory (RAM).

## Error Indicators

If the selected device operation is successfully executed for each device installed in the sockets, a completion message appears such as

```
        PROGRAM
SUMCHECK = 001AE5
```

A pass signal (two slow beeps) sounds and the LEDs below the sockets containing devices light green. However, if execution is unsuccessful for any device, a failure signal (three quick beeps) sounds, the LED below the failed device lights red, and a failure message appears as in the example below. (Refer to the Error Messages section at the back of the manual for a description of the error messages.)

```
        PROGRAM
BAD INSERTION 36
```

## Devices with Electronic Identifiers

To benefit from the electronic identifier feature built into many devices, an electronic identifier mode can be selected that causes the programmer to automatically select the correct programming algorithm and pin configuration. To select the electronic identifier mode when blank checking, loading, programming, or verifying a device, enter "FFFF" when prompted for the family and pinout code, or press a scroll key, scroll to the top of the device list, and select "Electronic ID."

When the electronic identifier mode is selected, the programmer determines the programming algorithm and pin configuration from the leftmost device that contains an electronic identifier. The programmer then checks to make sure that no other installed devices contain conflicting electronic identifiers. If a conflict between devices exists, the programmer displays an error message. Remove the conflicting device(s) and attempt the operation with only the compatible devices installed. If the electronic identifier mode is selected and no installed devices contain an electronic ID, an error message will be displayed and the electronic ID mode can not be used for the installed device(s). If an operation is completed successfully using the electronic ID mode, the electronic ID family and pinout codes become the default family and pinout codes.

| Procedure | Example 288A Displays |
|---|---|
| 1. Enter 0, A, F, 0, 3, 3 | |
| The display reads | INSERT<br>AF/33 |

The family/pinout code for each programmable device is contained in the Device List accompanying this manual. In the menus and also in the Device List, the device manufacturers are listed alphabetically. The parts for each device manufacturer are listed in order of device size, with multiple devices of the same size listed in numerical order.

## The Action Display

A special action display appears on the programmer during the execution of certain operations, such as self test, program, and verify. This display consists of decimal points advancing across the lower portion of the front panel display and indicates that the programmer is executing the operation.

## Aborting an Operation

Most operations may be aborted by pressing one of the hexadecimal keys. When an operation is halted in this manner, the programmer beeps and reverts to the currently selected mode; e.g., if a hexadecimal key is pressed during a "Load from Master" operation, the programmer beeps and the display returns to

```
LOAD FROM MASTER
```

If a hexadecimal key is accepted as input, press a scroll key to abort the operation.

| Procedure | Example 288A Displays |
|---|---|
| 6. Press ENTER to select the part number. | INSERT DEVICE<br>AMD 2764 |
| 7. Insert the device(s) into the socket(s), beginning at the leftmost socket, and press ENTER to initiate the check. | BLANK CHECK<br>. . . . . |
| If the blank check is successful for all of the installed devices, the LEDs below the sockets light green and the display reads | BLANK CHECK<br>OK |
| If the blank check operation is unsuccessful for any installed device, the LED for the failed device lights red and the display reads | BLANK CHECK<br>NONBLANK 20 |

8. Lift the socket lever(s) and remove the device(s) from the socket(s).

## CHECKING FOR NON-BLANK DEVICES

A blank check is provided that can be used prior to programming operations to insure that no data has been previously written to the device(s). The blank check verifies that no data exists at any address of any device installed in the programming sockets. Perform the blank check as follows:

*NOTE*

*The "288A Displays" shown in this manual are examples only. The devices shown in the displays may be different than the devices appearing on your printed Device List or in the menus on your unit since the device support capabilities of the 288A are updated periodically.*

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the BLANK CHECK function. | BLANK CHECK |
| 2. Press ENTER. | BLANK CHECK<br>F/P CODE FF/FF |
| 3. If the correct family/pinout code is displayed, press ENTER and go to step 7. If the correct code is not displayed, key in the 4-digit family/pinout code for the device (e.g., AF33), and skip to step 7<br><br>or<br><br>press a scroll key and then scroll to the device manufacturer (e.g., AMD). | BLANK CHECK<br>F/P CODE AF/33<br><br><br>BLANK CHECK<br>AMD |
| 4. Press ENTER when the correct manufacturer is displayed. | BLANK CHECK<br>AMD 2716 |
| 5. Scroll to the part number of the device (e.g., 2764). | BLANK CHECK<br>AMD 2764 |

## Loading the Data from a Master Device

The first step in programming a device is to load the data from the master device. This operation copies the programming data from the master device to the programmer's memory. When the transfer is complete, the programmer calculates and displays the sumcheck of the data. Use the following procedure to load the data from the master device.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the LOAD FROM MASTER function. | LOAD FROM MASTER |
| 2. Press ENTER. | LOAD FROM MASTER<br>F/P CODE FF/FF |
| 3. If the correct family/pinout code is displayed, press ENTER and go to step 7. If the correct code is not displayed, key in the 4-digit family/pinout code for the device (e.g., AF33), and skip to step 7<br><br>or<br><br>press a scroll key and then scroll to the device manufacturer (e.g., AMD). | LOAD FROM MASTER<br>F/P CODE AF/33<br><br><br>LOAD FROM MASTER<br>AMD |
| 4. Press ENTER when the correct manufacturer is displayed. | LOAD FROM MASTER<br>AMD 2716 |
| 5. Scroll to the part number of the device (e.g., 2764). | LOAD FROM MASTER<br>AMD 2764 |
| 6. Press ENTER to select the part number. | INSERT DEVICE<br>AMD 2764 |

## PROGRAMMING FROM A SINGLE MASTER

The 288A Multi Programmer can be used to program one or more devices with the data contained in a single master device, or with data downloaded to the programmer via the serial I/O (RS-232C) port. The following paragraphs describe the programming operation (for single devices and gangs) using a master device. If you choose to download the program data to the programmer's memory instead of loading from a master device, refer also to Downloading Data.

Programming a device consists of two basic operations:

- Loading the data from a master device to the programmer's memory (or downloading the data from a remote source such as a host computer — refer to Downloading Data).
- Programming the device (or devices) with the data copied to the programmer's memory.

## Programming Devices (Single or Gang)

After the master data has been copied or downloaded to the programmer, use the following procedure to program one or more devices with the data. The program operation copies the same block of data from the programmer's memory into each device installed in the programming sockets. When the programming operation is complete, the programmer calculates and displays the sumcheck of the data.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the PROGRAM function. | PROGRAM |
| 2. Press ENTER.<br><br>If the data to be programmed was just copied from the master device and the master is the same type of device as the device to be programmed, press ENTER and skip to step 7. Otherwise, proceed to select the device type for the part to be programmed. | PROGRAM<br>F/P CODE AF/33 |
| 3. Key in the 4-digit family/pinout code for the device (e.g., AF33) and skip to step 7<br><br>or<br><br>press a scroll key and then scroll to the device manufacturer (e.g., AMD). | PROGRAM<br>F/P CODE AF/33<br><br>PROGRAM<br>AMD |
| 4. Press ENTER when the correct manufacturer is displayed. | PROGRAM<br>AMD 2716 |
| 5. Scroll to the part number of the device (e.g., 2764). | PROGRAM<br>AMD 2764 |

| Procedure | Example 288A Displays |
|---|---|
| 7. Insert the master device into the socket and press ENTER to initiate the load operation. | LOAD FROM MASTER<br>. . . . . |
| If the load operation is successful, the LED below the socket containing the master device lights green. Note the sumcheck displayed on the programmer for later verification. | LOAD FROM MASTER<br>SUMCHECK = 00BA25 |
| If the load operation is unsuccessful, the LED below the socket containing the master device lights red and the programmer displays an error message. (See the Error Messages section for an explanation of the error message displayed.) | |
| 8. Lift the socket lever and remove the master device from the socket. | |

## PROGRAMMING FROM A SET OF MASTER DEVICES

When the length of a program exceeds the capacity of a single device, the program data must be partitioned into appropriately sized blocks and programmed into multiple devices; the size of each block is determined by the capacity of the device being used. See the following table for the block of programmer RAM assigned to each device in a set. The 288A automatically partitions the data and then programs the appropriate block of data into an individual device. The programmed devices comprise a "set" since, together, they contain the complete program data. The 288A can be used to program a set of devices (all of which use the same family/pinout codes) with the data contained in a set of master devices, or with data downloaded to the programmer via the serial I/O (RS-232C) port.

The number of unique devices in the set, called the "set size," can range from one to eight. You can produce multiple copies of the same set of devices in one Program operation by simply inserting each additional set of blank devices to be programmed into the sockets immediately following the previous set of blank devices. Each additional set will be programmed as a duplicate of the leftmost set of devices. You can also program partial sets in the same manner by inserting less than a complete set of blank devices.

The set programming feature can also be used to merge the data from a set of smaller devices into one or more larger ones. This function is performed in the same way as programming a set of devices except that the required number of larger blank devices are installed from left-to-right in the sockets and the appropriate set size is specified for the set of larger devices. The 288A automatically re-partitions the data and fills the first device. It then continues filling the rest of the devices with the remaining data.

The following paragraphs describe the set programming operation using a set of master devices. If you choose to download the program data to the programmer's memory instead of loading from a set of master devices, refer also to Downloading Data.

Programming a set of devices consists of two basic operations:

- Loading the data from a master set of devices into the programmer's memory (or downloading the data from a remote source such as a host computer—refer to Downloading Data).
- Programming the device set with the data copied to the programmer's memory.

| Procedure | Example 288A Displays |
|---|---|
| 6. Press ENTER to select the part number. | PROGRAM<br>SET SIZE = 1 |
| 7. If the displayed set size is 1, press ENTER to accept it. If the displayed set size is not 1, use the scroll keys to scroll to a set size of 1 and press ENTER to accept the new set size. The new set size becomes the default set size. A set size of 1 will cause the 288A to program all of the installed devices with the same block of programmer RAM data. The power-up default set size is 1. | INSERT DEVICE<br>AMD 2764 |
| 8. Insert the blank device(s) into any socket(s) and press ENTER to initiate the program operation. | PROGRAM<br>. . . . . |
| If the program operation is successful for all of the installed devices, the LEDs below the sockets light green. Note that the sumcheck displayed on the programmer matches that displayed at the end of the load operation. | PROGRAM<br>SUMCHECK = 00BA25 |
| If the program operation is unsuccessful for any installed device, the LED for the failed device lights red and an error message is displayed. (See the Error Messages section for an explanation of the error message displayed.) | |
| 9. Lift the socket lever(s) and remove the programmed device(s) from the socket(s). | |

| Socket Number | Device Size (no. of bits) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 16K | 32K | 64K | 128K | 256K | 512K | 1M | 2M | 4M |
| 1 | 000 | 000 | 0000 | 0000 | 0000 | 0000 | 00000 | 000000 | 0000000 |
| | 7FF | FFF | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFFF | 7FFFFFF |
| 2 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 | 400000 | 8000000 |
| | FFF | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF | 7FFFFF | FFFFFFF |
| 3 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 | 40000 | 800000 | 1000000 |
| | 17FF | 2FFF | 5FFF | BFFF | 17FFF | 2FFFF | 5FFFF | BFFFFF | 17FFFFF |
| 4 | 1800 | 3000 | 6000 | C000 | 18000 | 30000 | 60000 | C00000 | 1800000 |
| | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF | 7FFFF | FFFFFF | 1FFFFFF |
| 5 | 2000 | 4000 | 8000 | 10000 | 20000 | 40000 | 80000 | 100000 | |
| | 27FF | 4FFF | 9FFF | 13FFF | 27FFF | 4FFFF | 9FFFF | 13FFFF | |
| 6 | 2800 | 5000 | A000 | 14000 | 28000 | 50000 | A0000 | 140000 | |
| | 2FFF | 5FFF | BFFF | 17FFF | 2FFFF | 5FFFF | BFFFF | 17FFFF | |
| 7 | 3000 | 6000 | C000 | 18000 | 30000 | 60000 | C0000 | 180000 | |
| | 37FF | 6FFF | DFFF | 1BFFF | 37FFF | 6FFFF | DFFFF | 1BFFFF | |
| 8 | 3800 | 7000 | E000 | 1C000 | 38000 | 70000 | E0000 | 1C0000 | |
| | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF | 7FFFF | FFFFF | 1FFFFF | |

When loading data from more than one master into the programmer's memory, the number of master devices installed in the sockets is automatically detected, and data from sockets 1 through n is copied sequentially into the programmer's memory, where n is the highest socket number with an installed device. (The data bytes contained in the device installed in socket 1 are copied into RAM addresses in sequential order and are immediately followed by the data bytes contained in the device installed in socket 2 (in sequential order), and so on.) If any socket preceding the last device is left empty, "FF" is read into memory for the address range assigned to that socket.

The following table shows the maximum number of permissible masters for each device size, if the programmer has 2M of memory, and the address range in memory where data is copied from each socket.

| Procedure | Example 288A Displays |
|---|---|
| 7. Insert the master device set, beginning at the leftmost socket and press ENTER to initiate the load operation. | LOAD FROM SET<br>. . . . . |
| If the load operation is successful for all of the devices, the LEDs below the sockets containing devices light green. Note the sumcheck displayed on the programmer for later verification. | LOAD FROM SET<br>SUMCHECK = 005D2F |
| If the load operation is unsuccessful for any device installed, the LED for that socket lights red and the programmer displays an error message. (See the Error Messages section for an explanationof the error message displayed.) | |

8. Lift the socket levers and remove the master device set from the sockets.

| Procedure | Example 288A Displays |
|---|---|
| 6. Press ENTER to select the part number. | PROGRAM<br>SET SIZE = 1 |
| 7. If the displayed set size (number of devices in each programmed set) is correct, press ENTER to accept it. If the displayed number of devices in the set is not correct, use the scroll keys to scroll to the correct number and press ENTER. The new number will become the default set size. The power-up default set size is 1. | INSERT DEVICE<br>AMD 2764 |
| 8. Insert the blank devices into the sockets, beginning with the leftmost socket, and press ENTER to initiate the program operation. If you install more than one set of devices, the additional set(s) of devices will be programmed as duplicates of the first set. For example, if you specify a set size of 4 and install 8 devices, the devices in sockets 5, 6, 7 and 8 will be duplicates of the devices in sockets 1, 2, 3 and 4, respectively, after the program operation is complete. You can also program partial sets by inserting only a partial set of blank devices. | PROGRAM<br>. . . . . |
| If the program operation is successful for all of the installed devices, the LEDs below the sockets light green. Note that the sumcheck displayed on the programmer matches that displayed at the end of the load operation. | PROGRAM<br>SUMCHECK = 005D2F |
| If the program operation is unsuccessful for any installed device, the LED for the failed device lights red and an error message is displayed. (See the Error Messages section for an explanation of the error message displayed.) | |
| 9. Lift the socket levers and remove the programmed set(s) of devices from the sockets. | |

| Procedure | Example 288A Displays |
|---|---|
| 6. Press ENTER to select the part number. | PROGRAM<br>SET SIZE = 1 |
| 7. If the displayed set size (number of devices in each programmed set) is correct, press ENTER to accept it. If the displayed number of devices in the set is not correct, use the scroll keys to scroll to the correct number and press ENTER. The new number will become the default set size. The power-up default set size is 1. | INSERT DEVICE<br>AMD 2764 |
| 8. Insert the blank devices into the sockets, beginning with the leftmost socket, and press ENTER to initiate the program operation. If you install more than one set of devices, the additional set(s) of devices will be programmed as duplicates of the first set. For example, if you specify a set size of 4 and install 8 devices, the devices in sockets 5, 6, 7 and 8 will be duplicates of the devices in sockets 1, 2, 3 and 4, respectively, after the program operation is complete. You can also program partial sets by inserting only a partial set of blank devices. | PROGRAM<br>. . . . . |
| If the program operation is successful for all of the installed devices, the LEDs below the sockets light green. Note that the sumcheck displayed on the programmer matches that displayed at the end of the load operation. | PROGRAM<br>SUMCHECK = 005D2F |
| If the program operation is unsuccessful for any installed device, the LED for the failed device lights red and an error message is displayed. (See the Error Messages section for an explanation of the error message displayed.) | |
| 9. Lift the socket levers and remove the programmed set(s) of devices from the sockets. | |

## Programming a Set of Devices

The second step in programming one or more device sets is to program the blank device set(s) with the data copied to the programmer's memory. This operation copies the programming data from the programmer's memory to the device(s) installed in the programming socket(s), producing exact duplicates of the master set. When the programming operation is complete, the programmer calculates and displays the sumcheck of the data. Use the following procedure to copy the data from the programmer's memory to the set(s) of devices installed in the programming sockets.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the PROGRAM function. | PROGRAM |
| 2. Press ENTER.<br><br>If the data to be programmed was just copied from the master device set and the master set contains the same type of devices as the set to be programmed, press ENTER and skip to step 7. Otherwise, proceed to select the device type for the part to be programmed. | PROGRAM<br>F/P CODE AF/33 |
| 3. Key in the 4-digit family/pinout code for the device (e.g., AF33) and skip to step 7<br><br>or<br><br>press a scroll key and then scroll to the device manufacturer (e.g., AMD). | PROGRAM<br>F/P CODE AF/33<br>PROGRAM<br>AMD |
| 4. Press ENTER when the correct manufacturer is displayed. | PROGRAM<br>AMD 2716 |
| 5. Scroll to the part number of the device (e.g., 2764). | PROGRAM<br>AMD 2764 |

With the master data contained in the programmer's memory, perform the verify operation as follows:

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the VERIFY function. | VERIFY |
| 2. Press ENTER. If the data to be verified against was just copied from the master device or device set, and the master device or set is of the same type as the device or set to be verified, press ENTER and skip to step 7. Otherwise, proceed to select the device type for the part(s) to be verified. | VERIFY<br>F/P CODE AF/33 |
| 3. Key in the 4- or 6-digit family/pinout code for the device (e.g., AF33) and skip to step 7<br><br>or<br><br>press a scroll key and then scroll to the device manufacturer (e.g., AMD). | VERIFY<br>F/P CODE AF/33<br><br><br>VERIFY<br>AMD |
| 4. Press ENTER when the correct manufacturer is displayed. | VERIFY<br>AMD 2716 |
| 5. Scroll to the part number of the device (e.g., 2764). | VERIFY<br>AMD 2764 |
| 6. Press ENTER to select the part number. | INSERT DEVICE<br>AMD 2764 |

## VERIFYING PROGRAMMED PARTS

The verify operation allows you to check programmed devices to make sure that the data in the devices matches the data in the programmer's memory. The verify function compares device data with the data read into the programmer's memory during the preceding load or download operation.

The verify operation uses the Vcc high (VccH) and Vcc low (VccL) and number of passes recommended by the manufacturer of the device being verified. VccH and VccL levels can be selected when using Terminal Remote Control or Computer Remote Control.

If a "Load from Master" operation was performed to load the master data, then verification compares the data in each programmed device with the same memory data. (The verify operation is repeated for each device installed in the programmer's sockets.) If a "Load from Set" operation was performed, then verification compares each device in the set with the data stored in the associated address range of the programmer's memory.

Before you can verify the data within a device, a gang of devices, or a set of devices, you must first make sure that the master data is contained in the programmer's memory. You can load the master data into the programmer's memory by using one of the following methods:

- Perform the previously described Load from Master operation;
- Perform the previously described Load from Set operation;
- Perform a download operation (described later in this section).

| Procedure | Example 288A Displays |
|---|---|
| Pressing ENTER will display the high-byte data. | VERIFY<br>XXXXX HI R:XX D:XX<br>(addr)(RAM & device data) |
| Pressing ENTER advances to the next address where RAM data and device data don't match. When the programmer finishes verifying the device(s), the programmer will respond with a Verify Error Message. | VERIFY<br>VERIFY ERROR NN<br>(where NN is the error number) |

8. Lift the socket lever(s) and remove the verified device(s) from the socket(s).

| Procedure | Example 288A Displays |
|---|---|
| 7. Insert the device(s) to be verified into the socket(s) and press ENTER to initiate the verify operation. Insert the devices of a set in the same order that they were programmed in, so that each device is verified against the same block of programmer memory that it was originally programmed with. For example, to verify the first device of a two device set, insert the device into socket 1, 3, 5 or 7. | VERIFY<br>. . . . . |
| If the verify operation is successful for all of the installed devices, the LEDs light green and the sumcheck of the data is displayed. | VERIFY<br>SUMCHECK = 00BA25 |
| For unsuccessful verify operations, the programmer display differs for 8-bit devices and 16-bit devices as follows: | |
| If an 8-bit device verify operation is unsuccessful, the programmer will display the address location where the error occurs, the RAM data, and the device data. | VERIFY<br>XXXXX R:XX D:XX<br>(addr)(RAM & device data) |
| Pressing ENTER advances to the next address where RAM data and device data don't match. When the programmer finishes verifying the device(s), the programmer will respond with a Verify Error Message. | VERIFY<br>VERIFY ERROR NN<br>(where NN is the error number) |
| If a 16-bit device verify operation is unsuccessful, the 288A will display the address and the low-byte data. | VERIFY<br>XXXXX LO R:XX D:XX<br>(addr)(RAM & device data) |

## DOWNLOADING DATA

The 288A can receive and store data transferred from a host computer into its memory over a serial communications link. (Refer to the Getting Started section for information on serial I/O connections.) This feature may be used to download data from a computer, or other software development system, to the programmer in preparation for programming parts. The programmer can accept the data in any of the data translation formats shown on the 288A menus. Descriptions of each of these formats are provided in the Computer Remote Control section of this manual.

When initiating a download operation, the programmer prompts for the entry of three parameters in addition to the data translation format. These parameters are:

- the beginning address
- the block size
- the offset address

Selection of the beginning address allows you to make transfers of data to a specified start address in the programmer's memory. For example, if you specify a beginning address of 1000H (hexadecimal), the data transferred from the external data source will be placed in the programmer's memory starting at location 1000H instead of 0000H.

Selection of the block size allows you to specify how much data is to be transferred to the programmer's memory. For example, if you specify a block size of 2000H, the subsequent data transfer operation would store 8192 (2000H) bytes of data in the programmer's memory. To disable the selected block size and load data to the end of RAM (or the end of the data file, whichever is smaller), enter 0000H. The power-up default block size is 0000H.

## SETTING COMMUNICATIONS PROTOCOL

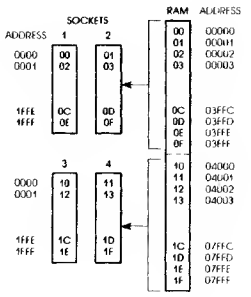
The programmer can be made to select the correct communications protocol automatically (see Terminal Remote Control or Computer Remote Control) or you can change the communications protocol from the power-up default values by using the following procedure. The power-up default values are:

baud rate = 9600 parity check = none number of data bits = 8 number of stop bits = 1

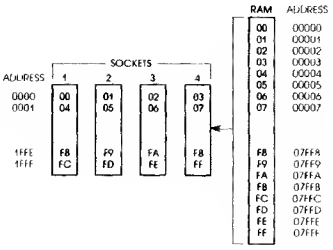
| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the RS232 PORT function. | RS232 PORT |
| 2. Press ENTER. | RS232 PORT<br>COMPUTER CONTROL |
| 3. Scroll to the PORT SETTING function. | RS232 PORT<br>PORT SETTING |
| 4. Press ENTER. | PORT SETTING<br>BAUD RATE = 9600 |
| 5. Scroll to the desired data transfer speed and press ENTER. | PORT SETTING<br>PARITY CHK.=NONE |
| 6. Scroll to the desired parity and press ENTER. | PORT SETTING<br>DATA BITS = 8 |
| 7. Scroll to the desired number of data bits and press ENTER. | PORT SETTING<br>STOP BIT = 1 |
| 8. Scroll to the desired number of stop bits and press ENTER. | RS232 PORT |

This completes the setting of the communications protocol for the serial I/O port.

**Programming of a set of four 64K bit devices using a 16-bit-wide word format:**




**Programming of a set of four 64K bit devices using a 32-bit-wide word format:**



Selection of the offset address allows you to specify the first data record of the data file in the host computer to be downloaded. Each data record in the data file contains a "record address" which specifies the address of the first data byte in the record. The record address is assigned when the program is compiled into formatted object code. During a download operation, the offset address specifies the record address of the first data byte that will be stored in RAM. For example, if you specify an offset address of 2000H, and a beginning RAM address of 1000H, the data bytes contained in the record with record address 2000H (the offset address) will be stored in RAM starting at the beginning RAM address (1000H). Data bytes contained in records with subsequent addresses will be stored in subsequent RAM addresses, up through the block size specified. Data records with record addresses less than the offset address will not be stored in RAM. The default offset address, "FFFFFFFF," has special meaning to the translators and causes the first data byte received to be stored in the beginning RAM address and subsequent data bytes received to be stored in subsequent RAM addresses.

When the downloading of the data is complete, you can set the word size of the data and the set size of the data for subsequent programming operations. You can select a word size of 8-bit-wide, 16-bit-wide or 32-bit wide. Using this feature, devices can be programmed with the downloaded data as single 8-bit-wide devices, as pairs of 16-bit-wide devices, or as sets of 32-bit-wide devices. The illustrations on the following page show the order in which RAM data is programmed into devices when a 16-bit-wide word size or a 32-bit-wide word size is selected.

| Procedure | Example 288A Displays |
|---|---|
| 8. Press ENTER to select the displayed offset address or key in the offset address (record address) of the first data record to be downloaded and press ENTER. (The default offset address, FFFFFFFF, will cause the first data byte received to be stored in the beginning RAM address.) | LOADING PORT |
| 9. Initiate the download of the data from the host computer.<br><br>When the download is complete, the sumcheck of the data received by the programmer will be displayed. | LOADING PORT<br>.....<br>LOADING COMPLETE<br>SUMCHECK = 009FF0 |
| 10. To return to the main menu and use the default word size (8-bit-wide), press ENTER or a hexadecimal keypad key. To set the word size to 16-bit-wide or 32-bit-wide and also select the appropriate set size for the word size specified, press a scroll key (as shown in the example display). | DOWNLOAD<br>WORD SIZE = 1 |
| 11. Select the word size using the scroll keys and press ENTER: 1 = 8-bit-wide word size, 2 = 16-bit-wide word size and 4 = 32-bit-wide word size. | DOWNLOAD<br>SET SIZE = 1 |
| 12. To select the displayed set size (number of complete sets to be programmed), press ENTER, or, to select a different set size, scroll to the desired set size and press ENTER. If you specified a word size of 2 (16-bit-wide), you can select a set size of 1, 2, 3, or 4. If you specified a word size of 4 (32-bit-wide), you can select a set size of 1 or 2. If you specified a word size of 1 (8-bit-wide), you can select any set size from 1 through 8. | RS232 PORT |

This completes the download operation.

being transferred. For example, if you specify a beginning RAM address of 8000H and an offset address of 2000H, then the data from programmer RAM address 8000H will be assigned a record address of 2000H. Consecutive RAM data bytes following the first data byte will be assigned consecutive record addresses (2001H, 2002H, and so on).

After hooking up the programmer to the host computer and setting the correct communications protocol, perform an upload operation as follows:

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the RS232 PORT function. | RS232 PORT |
| 2. Press ENTER. | RS232 PORT<br>COMPUTER CONTROL |
| 3. Scroll to the UPLOAD function. | RS232 PORT<br>UPLOAD |
| 4. Press ENTER. | UPLOAD FORMAT<br>INTEL INTELLEC 8 |
| 5. Scroll to the desired data translation format (e.g., Intel MCS-86 Hexadecimal) and press ENTER. | UPLOAD MCS-86<br>BE RAM AD=000000 |
| 6. Press ENTER to select the displayed beginning RAM address, or key in the hexadecimal beginning address of the data in the programmer's memory to be uploaded and press ENTER. | UPLOAD MCS-86<br>BLCK SIZE=000000 |

# UPLOADING DATA

The 288A can be used to transfer device data from its memory to a computer over a serial communications link. (Refer to the Getting Started section for information on serial I/O connections.) This feature may be used to upload data from the programmer's memory for the purposes of storing the data or editing it and then downloading it back to the programmer's memory. The programmer can transfer the data in any of the data translation formats shown on the 288A's menus. Descriptions of each of these formats are provided in the Computer Remote Control section of this manual.

When initiating an upload operation, the programmer prompts for the entry of three parameters in addition to the data translation format. These paramaters are:

- the beginning address
- the block size
- the offset address

Selection of the beginning address allows you to make transfers that use only a portion of data contained in the programmer's memory. For example, if you specify a beginning address of 8000H (hexadecimal), the data transfer operation will start with programmer RAM address 8000H (32,768 decimal) instead of the default beginning address. The power-up default beginning RAM address is 0000H.

Selection of the block size allows you to specify how much of the data contained in the programmer's memory is to be transferred, starting at the specified beginning address. For example, if you specify a beginning address of 8000H and a block size of 1000H, the subsequent data transfer operation would transfer only the data contained in addresses 8000H through 8FFFH. If you do not specify a block size, the default block size is the size of the current device times the set size specified in the last load or download operation.

Selection of the offset address allows you to select the address which will be assigned to the first data byte uploaded to the host computer. The data in programmer RAM is translated into the specified data translation format prinr to being sent through the RS-232C port. The data is formatted into data records which include a "record address." The record address is the address assigned to the first byte of data contained in that data record. The offset address specifies the first record address of the block of data

# EDITING RAM DATA

After the master program data has been loaded into the programmer's RAM, that data can be altered, added to, or deleted from using the editing functions. The data can also be searched through and some special programming features can be changed. The editing functions are summarized below. They are listed below in the same order that they occur in the 288A menus and are also discussed in more detail (in the same order) in the pages that follow.

- COMPLEMENT—inverts each byte of data in a range of RAM.
- INSERT—causes the programmer to enter insert mode and allow you to insert data bytes into consecutive addresses one-by-one. Existing data bytes are shifted up to higher addresses to accommodate the new data.
- BLOCK INSERT—inserts one value into a range of addresses. Existing data bytes are shifted up to higher addresses to accommodate the new data.
- DELETE—causes the programmer to enter delete mode and allow you to delete data bytes.
- BLOCK DELETE—deletes a block of data.
- BLOCK FILL—fills a range of addresses with a single value. The data originally contained in the filled addresses is overwritten.
- DATA SEARCH—searches for up to eight bytes of data in a specified range of RAM.
- BLOCK COPY—copies a block of data from one location in RAM to another, overwriting the data originally stored in the addresses copied to (destination addresses).
- DATA EDIT—enters an editing mode which allows you to change data bytes one-by-one.
- OPTION EDIT — allows you to change special programming functions, such as enabling and disabling electronic ID checking, specifying an initial byte, enabling or disabling synchronous programming (for Cypress devices), setting word sizes to 8, 16 or 32 bits and disabling the erase function for EEPROM devices.

| Procedure | Example 288A Displays |
|---|---|
| 7. Press ENTER to select the displayed block size, or key in the block size of the data to be uploaded and press ENTER. (Entering a block size of 0000 selects the current device size times the set size specified in the last load or download operation.) | UPLOAD MCS-86<br>OFFSET =FFFFFFFF |
| 8. Press ENTER to select the displayed offset address, or key in the offset address for the data and press ENTER. (The default offset address, FFFFFFFF, causes address 0000 to be assigned to the first data byte sent by the programmer.) | SENDING PORT<br>. . . . . |
| When the upload operation is complete, the programmer displays the sumcheck of the data that was sent over the serial RS-232C port in hexadecimal notation. | SENDING COMPLETE<br>SUMCHECK = 009FF0 |

## Complementing Data

Use the Complement function to invert the data contained in a specified range of memory (a 0 will become an F, an F will become a 0, an E will become a 1, a 1 will become an E, and so on). Data complements are based on 8-bit-wide word size.

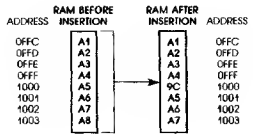
| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Press ENTER. | EDIT<br>COMPLEMENT |
| 3. Scroll to the COMPLEMENT function (if necessary) and press ENTER. | COMPLEMENT<br>START ADD=000000 |
| 4. Key in the first address of the range of data you want to complement and press ENTER. The default start address is 0. | COMPLEMENT<br>END ADD=07FFFF |
| 5. Key in the last address of the range of data you want to complement and press ENTER. The default end address is the end of memory (07FFFF for a programmer with 512K of memory). | COMPLEMENT<br>. . . . . |
| When the complement is complete, the display will read | COMPLEMENT<br>COMPLETED |

- BYTE SWAP—swaps adjacent bytes of data in a range of RAM.
- SHUFFLE—causes two equal size blocks of RAM to be shuffled into one large block of RAM with alternating bytes of data.
- SPLIT—causes one large block of RAM to split into two equal size blocks of RAM with alternating bytes of data.
- WORD EDIT—enters an editing mode which allows you to change data words one by one.

| Procedure | Example 288A Displays |
|---|---|
| 6. Press ENTER to insert the data into memory. When you press ENTER, the programmer will display the next consecutive address, at which the next keyed in byte of data will be inserted. To continue inserting data into consecutive addresses, return to step 5. To exit insert mode and return to the main menu, press either of the scroll keys. | ADDRESS = 001001<br>INSERT DATA = _ _ |

## Example

Insert data 9C at address 1000:



## Inserting Single Data Bytes

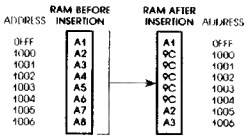
Use the Insert function to insert data bytes one at a time into consecutive addresses. (To insert the same value into a block of memory, use the Block Insert function.) Executing the Insert function causes the programmer to go into an insert mode. You may continue inserting data bytes into consecutive memory addresses until you press a scroll key. Pressing either of the scroll keys will cause the programmer to exit insert mode and return to the main menu. The original data at the address where new data is inserted is shifted up in memory (to higher addresses) to accommodate the new data. Data at the end of memory is lost.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | `EDIT` |
| 2. Press ENTER. | `EDIT`<br>`COMPLEMENT` |
| 3. Scroll to the INSERT function and press ENTER. | `INSERT`<br>`ADDRESS = 000000` |
| 4. Key in the address of the memory location where you want to begin inserting data (e.g., 1000) and press ENTER. The default start address of insertion is 0. | `ADDRESS = 001000`<br>`INSERT DATA = _ _` |
| 5. Key in the data value that you want to insert at the address displayed on the top line of the display. | `ADDRESS = 001000`<br>`INSERT DATA = 9C` |

| Procedure | Example 288A Displays |
|---|---|
| 7. Press ENTER to execute the Block Insert function. | BLOCK INSERT<br>. . . . . |
| When the insert function is complete, the display will read | BLOCK INSERT<br>COMPLETED |

### Example

Insert data 9C into block from 1000 through 1004:



## Inserting a Single Value into a Block of RAM

Use the Block Insert function to insert one value into a range of memory addresses. The original data in the range where new values are inserted is shifted up in memory (to higher addresses) to accommodate the new data. Data at the end of memory is lost. (To overwrite a range of memory addresses with a value, use the Block Fill function.)

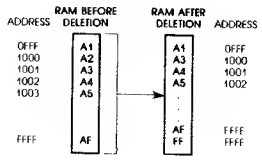
| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | `EDIT` |
| 2. Press ENTER. | `EDIT`<br>`COMPLEMENT` |
| 3. Scroll to the BLOCK INSERT function and press ENTER. | `BLOCK INSERT`<br>`START ADD=000000` |
| 4. Key in the first address of the range of memory in which data is to be inserted and press ENTER. The default start address is 0. | `BLOCK INSERT`<br>`END ADD=07FFFF` |
| 5. Key in the last address of the range of memory in which data is to be inserted and press ENTER. The default end address is the end of memory (07FFFF for a programmer with 512K of memory). | `BLOCK INSERT`<br>`DATA = _ __` |
| 6. Key in a one byte value to be inserted into the specified range of memory. | `BLOCK INSERT`<br>`DATA = 9C` |

| Procedure | Example 288A Displays |
|---|---|

6. To continue deleting successive data bytes, continue repeating step 5. To exit delete mode and return to the main menu, press either of the scroll keys.

## Example

Delete data at address 1000:



## Deleting Single Data Bytes

Use the Delete function to delete data bytes one at a time. Data following the deleted data byte is shifted down to fill in the deleted address and FF is filled in the last byte of RAM. Executing the Delete function causes the programmer to go into a delete mode. You may continue deleting successive data bytes until you press a scroll key. Pressing a scroll key causes the programmer to exit delete mode and return to the main menu.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | `EDIT` |
| 2. Press ENTER. | `EDIT`<br>`COMPLEMENT` |
| 3. Scroll to the DELETE function and press ENTER. | `DELETE`<br>`ADDRESS = 000000` |
| 4. Key in the address of the memory location where you want to begin deleting data (e.g., 1000) and press ENTER. The default start address of deletion is 0. The current value of the data contained in the specified address is displayed on the top line of the display. | `DELETE DATA = 93`<br>`ADDRESS = 001000` |
| 5. Press ENTER to delete the current contents of the displayed address. The next data byte is shifted down to the current address displayed. (The same address is displayed but it contains the next data byte.) | `DELETE DATA = 94`<br>`ADDRESS = 001000` |

## Filling a Block of Memory

Use the Block Fill function to overwrite the data contained in a range of RAM with a single value (data byte).

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Press ENTER. | EDIT<br>COMPLEMENT |
| 3. Scroll to the BLOCK FILL function and press ENTER. | BLOCK FILL<br>START ADD=000000 |
| 4. Key in the first address of the range of memory to be overwritten by the new data value (e.g., 1000) and press ENTER. The default start address is 0. | BLOCK FILL<br>END ADD=07FFFF |
| 5. Key in the last address of the range of memory to be overwritten by the new data value (e.g., 1FFF) and press ENTER. The default end address is the end of RAM (7FFFF for a programmer with 512K of RAM). | BLOCK FILL<br>DATA = _ _ |
| 6. Key in a one byte value to be written into the specified range of memory (e.g., 9C). | BLOCK FILL<br>DATA = 9C |
| 7. Press ENTER to execute the Block Fill function. | BLOCK FILL<br>. . . . . |
| When the Block Fill function is complete, the display will read | BLOCK FILL<br>COMPLETED |

## Deleting a Block of Data

Use the Block Delete function to delete a block of data stored in the programmer's RAM. Data following the deleted block is shifted down to fill in the deleted range and FF is filled in the vacated RAM addresses at the end of memory.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Press ENTER. | EDIT<br>COMPLEMENT |
| 3. Scroll to the BLOCK DELETE function and press ENTER. | BLOCK DELETE<br>START ADD=000000 |
| 4. Key in the address of the first data byte to be deleted (e.g., 1000) and press ENTER. The default start address is 0. | BLOCK DELETE<br>END ADD=07FFFF |
| 5. Key in the address of the last data byte to be deleted (e.g., 1FFF). The default end address is the end of RAM. | BLOCK DELETE<br>END ADD=001FFF |
| 6. Press ENTER to execute the Block Delete function. | BLOCK DELETE<br>. . . . . |
| When the delete function is complete, the display will read | BLOCK DELETE<br>COMPLETED |

| Procedure | Example 288A Displays |
|---|---|
| 7. To enter the next byte of data to be searched for, return to step 6, or press ENTER to execute the Data Search function for the data already entered. You may enter up to eight bytes of data to be searched for. After pressing ENTER to accept the eighth byte, the search will begin immediately and you will not have to press ENTER a second time. A match will occur only if the data is found in the same order as the typed-in string of data. | DATA = 01,02,03,<br>04,05,06,07,08 |
| 8. The programmer will search the specified range of memory for the data string entered on the display. If the data is found, the programmer will display the address of the first byte of the string found. You can then press ENTER to continue searching for other occurrences of the data in the specified range, or you can return to the main menu by pressing either scroll key. | DATA MATCH<br>ADDRESS = 10FF |
| If the data was not found in the specified range (or the data string was longer than the specified range) the programmer will display a fail message. Press any key to return to the main menu. | DATA SEARCH<br>FAIL |

## Searching for Data

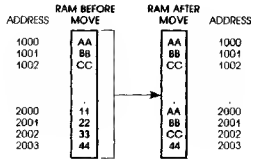
Use the Data Search function to search for up to eight bytes of data in any specified range of memory. You must specify a range of memory to search that is larger than the block of data bytes to be searched for or the data search will not be executed.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Press ENTER. | EDIT<br>COMPLEMENT |
| 3. Scroll to the DATA SEARCH function and press ENTER. | DATA SEARCH<br>START ADD=000000 |
| 4. Key in the address of the memory location where you want the data search to begin (e.g., 1000) and press ENTER. The default start address is 0. | DATA SEARCH<br>END ADD=07FFFF |
| 5. Key in the address of the memory location where you want the data search to end (e.g., 1FFF) and press ENTER. The default end address is the end of RAM (7FFFF for a programmer with 512K of RAM). | DATA = _ _ |
| 6. Key in the value of a byte to be searched for in the specified range of memory and press ENTER to enter that byte in the search string. (The comma will be provided by the programmer when you press ENTER.) | DATA = 01, |

| Procedure | Example 288A Displays |
|---|---|
| 7. Press ENTER to execute the Block Copy function. | BLOCK COPY<br>. . . . . |
| When the copy is complete, the display will read | BLOCK COPY<br>COMPLETED |

## Example

Copy block 1000 through 1002 to address 2000:



## Copying Data from One RAM Location to Another

Use the Block Copy function to copy data from one RAM location to another. The original data contained in the addresses to which the data was copied is overwritten by the copied data. Also make sure that the destination address of the block to be copied plus the block size of the block to be copied does not exceed programmer RAM (7FFFF for a programmer with 512K of memory).

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Press ENTER. | EDIT<br>COMPLEMENT |
| 3. Scroll to the BLOCK COPY function and press ENTER. | BLOCK COPY<br>START ADD=000000 |
| 4. Key in the address of the first data byte you want to copy and press ENTER. The default start address is 0. | BLOCK COPY<br>END ADD=07FFFF |
| 5. Key in the address of the last byte of data you want to copy and press ENTER. (If you want to copy only one byte of data, type in the same value for the end address as you did for the start address.) The default end address is the end of RAM. | BLOCK COPY<br>DESTN ADD=000000 |
| 6. Key in the address to which the first byte of data will be copied (destination address) (e.g., 2000). The rest of the data to be copied will be copied into consecutive addresses following the destination address. | BLOCK COPY<br>DESTN ADD=002000 |

## Swapping Adjacent Bytes of Data

Use the Byte Swap function to swap adjacent bytes of data through a range of RAM.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Press ENTER. | EDIT<br>COMPLEMENT |
| 3. Scroll to the BYTE SWAP function and press ENTER. | BYTE SWAP<br>START ADD = 000000 |
| 4. Key in the address of the first data byte you want swapped and press ENTER. The default start address is 0. | BYTE SWAP<br>END ADD = 1FFFFF |
| 5. Key in the address of the last data byte you want swapped (the block size must be even). The default end address is the end of RAM. | |
| Press ENTER to execute the BYTE SWAP function. | BYTE SWAP<br>.... |
| When the BYTE SWAP function is complete, the display will read | BYTE SWAP<br>COMPLETED |

## Editing Single Data Bytes

Use the Data Edit function to edit individual bytes of data stored in the programmer's RAM. Executing this command causes the programmer to go into a special edit mode that will allow you to display and edit data bytes one-by-one until you press a scroll key.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | `EDIT` |
| 2. Press ENTER. | `EDIT`<br>`COMPLEMENT` |
| 3. Scroll to the DATA EDIT function and press ENTER. | `DATA EDIT`<br>`ADDRESS = 000000` |
| 4. Key in the address of the first data byte you want to edit (e.g., 1000) and press ENTER. The default address to start editing at is 0. The current value of the data at the address displayed is shown on the second line of the display (e.g., 05). | `ADDRESS = 001000`<br>`DATA 05 --> _ _` |
| 5. To change the current data byte (as in the example), key in the new value and press ENTER. To leave the current data byte unchanged and advance to the next byte to be edited, press ENTER without keying in a new value. | `ADDRESS = 001000`<br>`DATA 05 --> 9C` |
| 6. After pressing ENTER, the next address will be displayed on the top line of the display and the current value of the data at that address will be displayed on the second line of the display. To edit the displayed data byte, repeat step 5. To exit the edit mode and return to the main menu, press either scroll key. If you key in a new value and then press a scroll key before pressing ENTER, the keyed in data will be ignored and the current data byte will remain unchanged. | `ADDRESS = 001001`<br>`DATA 06 --> _ _` |

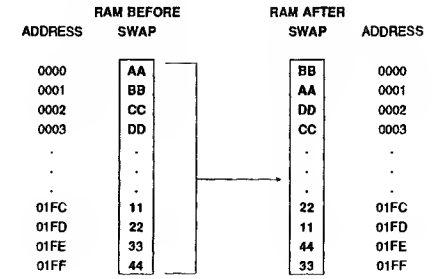
## Shuffling Two Blocks of RAM Together

Use the Shuffle function to shuffle two equal size blocks of RAM together to form one large block of RAM. The large block of RAM will contain alternating bytes of data from the two original blocks of RAM.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | `EDIT` |
| 2. Press ENTER. | `EDIT`<br>`COMPLEMENT` |
| 3. Scroll to the SHUFFLE function and press ENTER. | `SHUFFLE`<br>`MID ADR = 100000` |
| 4. Key in the address of the second block of data (the block size must be even) and press ENTER. The default middle address is the middle of the RAM. | |
| Press ENTER to execute the SHUFFLE function. | `SHUFFLE`<br>`.....` |
| When the SHUFFLE function is done, the display will read | `SHUFFLE`<br>`COMPLETED` |

### Example

Swap bytes from address 0000 through address 01FF:



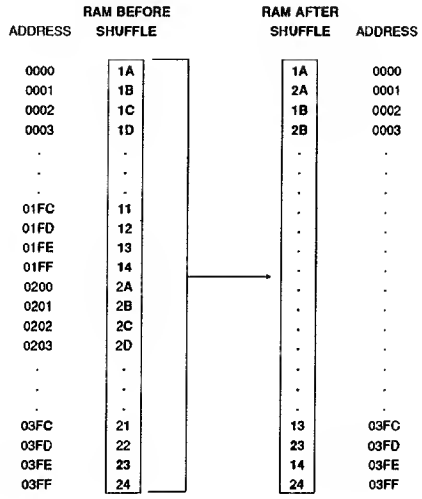
## Splitting into Two Blocks of RAM

Use the Split function to split a large block of RAM into two equal size blocks of RAM. The large block of RAM will alternate bytes of data into the two smaller blocks.

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Press ENTER | EDIT<br>COMPLEMENT |
| 3. Scroll to the SPLIT function and press ENTER. | SPLIT<br>MID ADR = 100000 |
| 4. Key in the address of the middle of the block of data (the block must be even) and press ENTER. The default middle address is the middle of the RAM. | |
| Press ENTER to execute the SPLIT function. | SPLIT<br>..... |
| When the SPLIT function is done, the display will read | SPLIT<br>COMPLETED |

### Example

Shuffle data bytes at a middle address of 0200:



| ADDRESS | RAM BEFORE SHUFFLE | RAM AFTER SHUFFLE | ADDRESS |
|---|---|---|---|
| 0000 | 1A | 1A | 0000 |
| 0001 | 1B | 2A | 0001 |
| 0002 | 1C | 1B | 0002 |
| 0003 | 1D | 2B | 0003 |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| 01FC | 11 | . | . |
| 01FD | 12 | . | . |
| 01FE | 13 | . | . |
| 01FF | 14 | . | . |
| 0200 | 2A | . | . |
| 0201 | 2B | . | . |
| 0202 | 2C | . | . |
| 0203 | 2D | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| 03FC | 21 | 13 | 03FC |
| 03FD | 22 | 23 | 03FD |
| 03FE | 23 | 14 | 03FE |
| 03FF | 24 | 24 | 03FF |

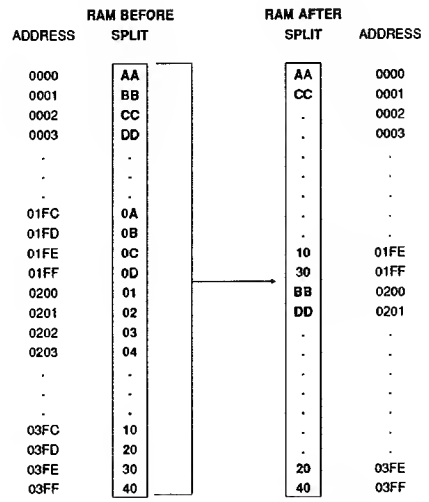
## Editing Data Words

Use the Word Edit function to edit words of data stored in the programmer's RAM. Executing this command causes the programmer to go into a special edit mode that will allow you to display and edit data words one-by-one until you press a scroll key.

| Example Procedure | 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Press ENTER. | EDIT<br>COMPLEMENT |
| 3. Scroll to the WORD EDIT function and press ENTER. | WORD EDIT<br>ADDRESS = 000000 |
| 4. Key in the address of the first data byte you want to edit (e.g., 1000) and press ENTER. The default address to start editing at is 0. The current value of the data at the address displayed is shown on the second line of the display (e.g., 0504). | ADDRESS = 001000<br>DATA 0504 --> ____ |
| 5. To change the current data byte (as in the example), key in the new value and press ENTER. To leave the current data byte unchanged and advance to the next byte to be edited, press ENTER without keying in a new value. | ADDRESS = 001000<br>DATA 0504--> 9C8C |
| 6. After pressing ENTER, the next address will be displayed on the top line of the display and the current value of the data at that address will be displayed on the second line of the display. To edit the displayed data word, repeat step 5. To exit the edit mode and return to the main menu, press either scroll key. If you key in a new value and then press a scroll key before pressing ENTER, the keyed in data will be ignored and the current data byte will remain unchanged. | ADDRESS = 001002<br>DATA 0607 --> ____ |

### Example

Split data bytes at a middle address of 0200:



| Procedure | Example 288A Displays |
|---|---|
| 5. To change the current status of electronic ID checking, press a scroll key and then press ENTER. To leave the current status unchanged, press ENTER without pressing a scroll key, or scroll back to the previous value and press ENTER. | ID EN/DIS<br>DISABLE |

## Setting the Initial Byte

To set the value of the initial byte to be programmed into devices supporting the initial byte feature, perform the following procedure:

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Press ENTER. | EDIT<br>COMPLEMENT |
| 3. Scroll to the OPTION EDIT function and press ENTER. | OPTION EDIT<br>ID EN/DIS |
| 4. Scroll to the INITIAL BYTE function and press ENTER. | INITIAL BYTE<br>DATA 00 --> _ |
| 5. Type in the value of the initial byte data (e.g., B5) and press ENTER. To leave the value unchanged, press a scroll key. The default data value is 00 (zero). | INITIAL BYTE<br>DATA 00 --> B5 |

## Changing Special Programming Options

Use the Option Edit function to change the special programming functions. For instance, use this function to enable or disable electronic ID checking. If electronic ID checking is enabled, the 288A will check to make sure that the installed device's electronic ID matches the device type selected during a device operation (such as programming). If a mismatch occurs, an error message will be displayed and the operation will be halted. Option Edit also allows you to use special features available on Cypress devices. These features are initial byte programming and synchronous programming. See the device manufacturer's documentation for information on these device features. Option Edit also allows you to set the word size to 8, 16, or 32 bits and disable the erase function for EEPROM devices.

### Enabling/Disabling Electronic ID Checking

To enable or disable electronic ID checking, perform the following procedure:

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | `EDIT` |
| 2. Press ENTER. | `EDIT`<br>`COMPLEMENT` |
| 3. Scroll to the OPTION EDIT function and press ENTER. | `OPTION EDIT`<br>`ID EN/DIS` |
| 4. Scroll to the ID ENABLE/DISABLE function (if necessary) and press ENTER. If electronic ID checking is currently enabled "ENABLE" will appear on the lower portion of the display. If electronic ID checking is disabled, "DISABLE" will appear on the display. | `ID EN/DIS`<br>`ENABLE` |

## Setting Word Size

The Factory Default Word Size is 8 bits. By using a scroll key, you can select a word size of 8, 16, or 32 bits. To change the word size setting, perform the following procedure:

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | `EDIT` |
| 2. Scroll to the OPTION EDIT function and press ENTER. | `OPTION EDIT`<br>`WORD SIZE` |
| 3. Scroll to the WORD SIZE function and press ENTER. If the Factory Default is enabled, "8 BITS" will appear on the lower portion of the display. | `WORD SIZE`<br>`8 BITS` |
| 4. Use a scroll key to scroll from 8 bits to 16 bits to 32 bits. Press ENTER to choose the desired selection. | `WORD SIZE`<br>`16 BITS` |

## Enabling/Disabling Synchronous Programming

To enable or disable synchronous programming of devices which support this feature, perform the following procedure:

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | `EDIT` |
| 2. Press ENTER. | `EDIT`<br>`COMPLEMENT` |
| 3. Scroll to the OPTION EDIT function and press ENTER. | `OPTION EDIT`<br>`ID EN/DIS` |
| 4. Scroll to the SYNCH EN/DIS function and press ENTER. If synchronous programming is currently disabled "DISABLE" will appear on the lower portion of the display. If synchronous programming is enabled, "ENABLE" will appear on the display. Synchronous programming is disabled automatically on power up. | `SYNCH EN/DIS`<br>`DISABLE` |
| 5. To change the current status of synchronous programming, press a scroll key and then press ENTER. To leave the current status unchanged, press ENTER without pressing a scroll key, or scroll back to the previous value and press ENTER. | `SYNCH EN/DIS`<br>`ENABLE` |

# 3. TERMINAL REMOTE CONTROL

This section explains how to operate the 288A Multi Programmer from a remote terminal. Most of the operations that can be performed from the front panel are available in Terminal Remote Control (TRC) mode, as well as sumchecking commands, and commands for programming with data in 16-bit-wide word format and 32-bit-wide word format.

The commands for operation from a remote terminal are grouped into the following subsections:

- INTRODUCTION—This section includes a list of the symbols and conventions used in this section of the manual, a quick reference summary of all of the commands available in Terminal Remote Control, and some general operating notes. Read the general operating notes before attempting any of the Terminal Remote Control Commands.
- ENTERING AND EXITING TERMINAL REMOTE CONTROL—This subsection explains how to transfer control of the programmer to a remote terminal and how to return control of the programmer to the front panel.
- ON-LINE HELP—This subsection explains how to display the on-line help screen and how to read the help screen symbols.
- SELECTING A DEVICE TYPE—This subsection explains how to select the device type using the 288A TRC menus or by entering the family/pinout code directly. The correct device type must be selected prior to performing programming operations, otherwise damage to the installed devices could result. This section also explains how to enable electronic ID checking. This feature causes the 288A to read the installed device's electronic ID and make sure that the family/pinout code selected is correct.
- PROGRAMMING OPERATIONS—This subsection explains how to perform a blank check on devices prior to programming them, load the master data into RAM from previously programmed devices, and program blank devices with the master data.
- VERIFYING PROGRAMMED DEVICES—This subsection explains how to verify that the data programmed into the installed devices matches the data in the programmer's RAM. Devices are automatically verified during the programming operation, but this command allows you to verify the devices at another time.

## Enabling/Disabling the Erase Function

The Erase En/Dis function allows you to disable the Erase function so that you may change some of the programming function of an EEPROM device without erasing all of the programming information contained in that device. By disabling the Erase function, for instance, you could change a location only and leave all of the other information contained in that device intact. To toggle between the En/Dis Erase function, perform the following procedure:

| Procedure | Example 288A Displays |
|---|---|
| 1. Scroll to the EDIT function. | EDIT |
| 2. Scroll to the OPTION EDIT function and press ENTER. | OPTION EDIT<br>ERASE EN/DIS |
| 3. Scroll to the ERASE EN/DIS function and press ENTER. | ERASE EN/DIS<br>DISABLED |
| 4. Use a scroll key to toggle between DISABLE and ENABLE. Press ENTER to choose the desired selection. | ERASE EN/DIS<br>ENABLED |

# INTRODUCTION

## Symbols and Conventions

The following is a list of symbols and conventions used in the Terminal Remote Control section of this document:

| | |
|---|---|
| <CR> | This symbol represents the carriage return key. |
| <ESC> | This symbol represents the escape key. |
| <CTRL-S> | CTRL- followed by a letter means that you should hold down the CONTROL (or CTRL) key and press the letter once. For example, to type <CTRL-S>, you would hold down the CONTROL key and press the S key once. |
| H | Any value followed by an upper-case H means that the value is in hexadecimal form. |
| <lower-case> | Lower-case letters enclosed in angle brackets describe the kind of parameter that you should fill in for the command. Replace the words and the brackets with the appropriate parameter value for the operation you are performing. For example, the Memory Modify command format is M <start address>. You would enter M 6 to modify memory starting at hexadecimal address 6. |
| [<lower-case>] | Words or characters enclosed in square braces represent optional entries. |
| word-wide | Refers to data formatted in 16-bit-wide words. |
| long-word-wide | Refers to data formatted in 32-bit-wide words. |

- EDITING MEMORY—This subsection explains how to edit the contents of the programmer's data memory, or RAM.
- SUMCHECKING DATA—This subsection explains how to perform a sumcheck (hexadecimal summation) and a checksum (exclusive-OR summation) on the data stored in the programmer's RAM.

## General Operating Notes

This subsection contains important information on how to type in the Terminal Remote Control (TRC) commands, as well as information on how to use special keys to control command execution and how to read the device error indicators. A list of the default parameter values for the TRC commands is given at the end of this subsection.

### Entering Commands

Below are some general rules for entering Terminal Remote Control commands.

- Each command must be entered at the command prompt (>) and must be followed by a <CR>.
- All commands must be typed in upper-case letters.
- In most cases, the first parameter (such as a memory address) that follows the command letter(s) can either be separated from the command letter(s) by a space or it can be typed immediately following the command letter(s) with no space. For example, the Load Set command could be typed LS4 or LS 4 to load a set of four masters. The one exception is the command to verify a single device (or gangs of identical devices), which requires a space between the command letter (V), and the Vcc selection (L or H) in order to distinguish it from the Verify Long Word command (VL).
- Each parameter following the first parameter must be separated from the preceding parameter by a space. For example, to display memory addresses 0 through F, the Display Memory command must be typed M 0 F or M0 F.
- You do not need to enter leading zeros when entering command parameters, such as memory addresses. For example, 0010 is the same as 10.
- All command parameters are entered as hexadecimal numbers, unless otherwise specified.

## Terminal Remote Control Command Summary

| Command Format | Description |
|---|---|
| M | Enter TRC mode |
| <CR> | Execute command |
| <ESC> | Abort command |
| <BS> | Backspace (key) |
| <CTRL-S> | Suspend display |
| <CTRL-Q> | Continue display |
| A? | Select device type |
| B | Blank check installed devices |
| L [a1 a2 a3] | Load single master |
| LS [set no.] | Load Set of masters |
| LW [set no.] | Load Word-wide masters |
| LL [set no.] | Load Long-word-wide masters |
| P [a1 a2 a3] | Program single or gangs |
| PS [set no.] | Program Set of devices |
| PW [set no.] | Program Word-wide set |
| PL [set no.] | Program Long-word-wide set |
| V [Vcc] [a1 a2 a3] | Verify single or gangs |
| VS [Vcc] [set no.] | Verify Set of devices |
| VW [Vcc ] [set no.] | Verify Word-wide set |
| VL [Vcc] [set no.] | Verify Long-word-wide set |
| ? | Display help screen |
| CT a1 a2 [a3] | Sumcheck (Total) |
| CX a1 a2 [a3] | Checksum (EXQR) |
| M a1 a2 | Display Memory |
| M a1 | Modify Memory |
| M a1 a2 d1 [d2...d8] | Fill Memory |
| MW a1 a2 | Display Memory Word |
| MW a1 | Modify Memory Word |
| MW a1 a2 d1 [d2...d8] | Fill Memory Word |
| I a1 d1 [d2...d8] | Insert data |
| D a1 [a2] | Delete data |
| T a1 a2 a3 | Transfer (copy) data |
| MS a1 a2 d1 [d2...d8] | Search Memory |
| BS a1 a2 | Byte Swap |
| ID (1/0) | Enable ID |
| EO (1/0) | Erase Option |
| R | Exit TRC mode (Return) |
| S ff/pp | Select family/pinout code |

### List of Symbols

[ ] indicates optional entry
a1 = starting RAM address
a2 = ending RAM address
a3 = starting device or destination address
d1 d2...d8 = data bytes 1 through 8
1/0 = on/off selection (1=on, 0=off)
ff/pp = four digit family/pinout code
set no. = set size
Vcc = Vcc selection (H or L)

## Error Indicators

The LED below each of the sockets is used to indicate successful or unsuccessful programming operations. Many Terminal Remote Control commands also cause the programmer to display a table showing the results of the operation performed on each socket (see the example below).

```
DEVICE: 1 2 3 4 5 6 7 8
        P P P F X X X X
```

If an operation is completed successfully on a device, a "P" (which stands for "pass") will appear below the socket number (see example above) containing that device and the LED below the socket will light green.

If an operation is not completed successfully on a device, an "F" (which stands for "fail") will appear below the socket number (see example above) containing that device and the LED below the socket will light red.

If a socket is left empty during an operation, an "X" will appear below the socket number of the empty socket (see example above) and the LED below the socket will remain off.

A "SYNTAX ERROR" will occur when the format of a command is incorrect or when illegal characters are entered in a command. A caret (^) will appear below the first unrecognized or illegal character encountered by the programmer.

## Controlling Command Execution

The following keys are special keys which can be used to control the execution of TRC commands.

*BACKSPACE Key*
While you are entering TRC commands, you can use the <BS> key (or <Delete>, <Rubout>, or <CTRL-H>, if your terminal keyboard has no <BS> key) to delete the previously entered character. You can use this key repeatedly until you have deleted the entire command line.

*ESCAPE Key*
You can abort the command currently being executed or the command you are currently typing by pressing the <ESC> key. If you press the <ESC> key while typing in a command (before pressing <CR>), the programmer will ignore the line you were typing, display the command prompt (>) on a new line, and await your input. If you press the <ESC> key while a command is being executed, the programmer will abort the command, display the command prompt (>), and await your input.

*<CTRL-S>*
Typing a <CTRL-S> will suspend the displaying of data on the terminal screen. The command that was being executed when the <CTRL-S> was typed is temporarily halted, but not aborted. To continue the display, type <CTRL-Q>.

*<CTRL-Q>*
Typing a <CTRL-Q> continues the displaying of data on the terminal screen exactly where it was suspended by the <CTRL-S> command.

## Action Display

While an operation is being executed in TRC mode, advancing dots will appear on the bottom line of the programmer display, indicating that the programmer is performing the operation.

## Default Parameter Values

Many of the Terminal Remote Control commands can be accompanied by optional parameters, such as starting and ending memory addresses. The power-up default values of these optional parameters are listed below. When a new set size is entered or a new device type is selected, the new value becomes the default for future operations. The default values of the <start address>, <end address>, <device address>, and <Vcc> will remain the same as listed below even if a new value has been specified in a previous command.

| | |
|---|---|
| <start address> | 0000H |
| <end address> | 1FFFH or the size of the selected device for device operations |
| <device address> | 0000H |
| <set size> | 1 |
| <Vcc> | for verify operations device manufacturer's recommendation |
| Device type | reads Electronic ID of installed device (initial family/pinout code is FF/FF) |

## Exiting Terminal Remote Control

You can exit Terminal Remote Control mode by using either the terminal keys or the programmer keys. To exit TRC using the terminal keys, type

R<CR>

(The R command stands for Return to local.) To exit TRC mode using the programmer keys, press any key.

The programmer will display

```
RS232 PORT
```

You can now continue operating the programmer using the front panel keys.

| Procedure | Example 288A Displays |
|---|---|
| 5. To establish communications between the programmer and the terminal and automatically select the proper communications protocol, type<br><br>M<CR><br><br>on the terminal keyboard. (The "M" will not be displayed on the terminal screen.) You can also set the communications protocol manually, by executing the PORT SETTING function (see the Front Panel Operation section). If the correct port settings are selected prior to entering TRC mode, you can press any terminal key to establish communications.<br><br>When communication is established, a command prompt (>) will appear on the terminal screen. The programmer will display TERMINAL CONTROL until control of the programmer is returned to the front panel. | TERMINAL CONTROL |

## SELECTING A DEVICE TYPE

Use the commands described in the following paragraphs to select the type of the device you are loading from, blank checking, programming or verifying. The device type can be selected in one of two ways — by selecting the correct manufacturer and part number from the TRC menus or by entering the Data I/O family/pinout code directly. Selecting the device type tells the programmer which algorithm and voltages to use to program the part. ***You must select the correct device type before blank checking, loading, programming, or verifying a device, otherwise damage to the device could result. The 288A also allows you to make use of the electronic ID contained in many devices.*** When a family/pinout code of FFFF is selected, the 288A will read the electronic ID of the installed device and automatically select the correct family/pinout code for the device. You may also enable the Electronic ID checking feature (explained in this subsection), which causes the 288A to display an error message if the selected device type and the electronic ID contained in the installed device do not match. The current device type and family/pinout code selected can be viewed by typing ?<CR>.

## ON-LINE HELP

A help screen displaying the currently selected device type and a list of all of the TRC commands and their formats can be displayed on the terminal screen by typing

?<CR>

at the command prompt (>). The help screen is shown below:

```
          *** 288A MULTI PROGRAMMER COMMAND HELP ***

SELECT DEVICE TYPE A?              \   SUMCHECK (TOTAL)          CT a1 a2 [a3]
BLANK CHECK        B               \   CHECKSUM (EXOR)          CX a1 a2 [a3]
LOAD SINGLE        L [a1 a2 a3]    \   MEMORY DISPLAY (WORD)    M (MW) a1 a2
LOAD SET           LS [ # ]        \   MEMORY MODIFY (WORD)    M (MW) a1
LOAD WORD          LW [ # ]        \   MEMORY FILL (WORD)      M (MW) a1 a2 d1[..d8]
LOAD LONG WORD     LL [ # ]        \   INSERT                  I a1 d1[..d8]
PROGRAM SINGLE     P [a1 a2 a3]    \   DELETE                  D a1 [a2]
PROGRAM SET        PS [ # ]        \   TRANSFER                T a1 a2 a3
PROGRAM WORD       PW [ # ]        \   MEMORY SEARCH           MS a1 a2 d1[..d8]
PROGRAM LONG WORD  PL [ # ]        \   BYTE SWAP               BS a1 a2
VERIFY SINGLE      V [H/L][a1 a2 a3] \ ENABLE ID              ID (1/0)
VERIFY SET         VS [H/L][ # ]   \   ERASE OPTION           EO (1/0)
VERIFY WORD        VW [H/L][ # ]   \   RETURN TO LOCAL        R
VERIFY LONG WORD   VL [H/L][ # ]   \   SET F/P CODE           S code
HELP               ?

DEVICE TYPE: AMD 2716     F/P: 19/23          SETSIZE #: 1
ELECTRONIC ID: ENABLE     ELECTRICAL ERASE:   ENABLE
```

### List of Symbols

[ ] indicates optional entry
a1 = starting RAM address
a2 = ending RAM address
a3 = starting device or destination address
d1..d8 = data bytes 1 through 8
# = set size
H/L = Vcc selection (H or L)
1/0 = on/off selection (1=on, 0=off)
ff/pp = four digit family/pinout code

**Example**

Select Intel device 2764:

```
>A?

DEVICE MANUFACTURERS
0.AMD                 8.INTEL           16.OKI            24.TEXAS INSTRUMENT
1.ATMEL               9.ICT             17.RICOH          25.TOSHIBA
2.CYPRESS             10.MATSUSHITA     18.ROCKWELL       26.TRISTAR SEMICON
3.EUROTECHNIQUE       11.MITSUBISHI     19.SAMSUNG        27.VTI
4.EXEL                12.MOSTEK         20.SEEQ           28.XICOR
5.FUJITSU             13.MOTOROLA       21.SGS-ATES
6.GENERAL INSTRU.     14.NATIONAL       22.SIGNETICS
7.HITACHI             15.NEC            23.SMOS SYES

ENTER MANUFACTURER NUMBER AND RETURN 8<CR>

INTEL DEVICES
0. 2716     5. 2732B    10. P2764       15. 27256
1. 2816     6. P2732A   11. P2764A      16. 27C256
2. 2816A    7. 2764     12. 27128       17. 27512
3. 2732     8. 2764A    13. 27128A      18. 27513
4. 2732A    9. 27C64    14. P27128A

ENTER PART NUMBER AND RETURN 7<CR>

FAMILY AND PINOUT CODE IS 79/33
```

*NOTE*

*The above list is only an example. The number of devices available at the time you purchase your programmer may be different.*

## Selecting the Device Type from the TRC Menus

Use the Select Device Type command to select the manufacturer and part number of the device you are loading from, blank checking, programming, or verifying from the TRC menus. You do not need to know the device's assigned family/pinout code to select the device type using this command. After you have selected the device's manufacturer and part number using this command, the 288A Multi Programmer will automatically supply the proper family and pinout codes for the device.

Command Format: A?

| Procedure | Example Key Sequence |
|---|---|
| 1. Type in the Select Device Type command letters, A?, and press carriage return. | A?<CR> |
| 2. After the list of device manufacturers is displayed on the screen, type in the number that corresponds to the manufacturer of the device you are blank checking, loading, programming or verifying and press carriage return. If you press carriage return without entering a number, device manufacturer 0 will be selected. | 8<CR> |
| 3. After the list of device part numbers is displayed on the screen, type in the number that corresponds to the correct device part number and press carriage return. If you press carriage return without entering a number, part number 0 will be selected. The device type selected will remain the default device type until a new device type is selected. | 7<CR> |

## Enabling/Disabling Electronic ID Checking

Use the Electronic ID Checking feature to cause the programmer to read the electronic ID contained in the installed device(s) and make sure that the device type specified by the electronic ID matches the device type you selected. If this feature is enabled, the programmer will read the electronic ID of the installed device(s) at the beginning of a blank check, load, program, or verify operation. If the device type you selected does not match the installed device(s) electronic ID type, the programmer will not allow the operation to be performed. Electronic ID checking is enabled upon power up.

| Procedure | Example Key Sequence |
|---|---|
| 1. Type in the Electronic ID Checking command letters, ID, followed by a space. | `ID` |
| 2. If you want to enable the Electronic ID Checking feature, type a 1 and press carriage return. If you want to disable the Electronic ID Checking feature, type a 0 (zero) and press carriage return.<br><br>The programmer will display a message acknowledging that the Electronic ID Checking feature has either been enabled or disabled, depending upon whether you entered a 0 or a 1. | `ID 1<CR>`<br>or<br>`ID 0<CR>` |

## Selecting the Family/Pinout Code

Use the Set Family/Pinout Code command to select the correct device type by entering the Data I/O family/pinout code. The correct family/pinout code for each device programmable by the 288A is printed in the device list shipped with your programmer or module. Perform the following procedure to select the family/pinout code.

Command Format: S <ffpp>

| Procedure | Example Key Sequence |
|---|---|
| 1. Type in the Set Family/Pinout Code command letter, S, followed by a space. | S |
| 2. Type in the four-digit family/pinout code printed in the device list for the device you will be loading, programming, or verifying and then press carriage return.<br><br>*NOTE*<br><br>*If you type an S followed by a carriage return (without entering a family/pinout code), the programmer will prompt you to enter the family/pinout code.*<br><br>After you enter the family/pinout code, the programmer will return a message indicating whether the family/pinout code combination you entered was a legal code or was an illegal combination. If you receive a failure message, check your device list for the correct code and try entering it again. | S 7933<CR><br>or<br>S<CR><br>7933<CR> |

## PROGRAMMING OPERATIONS

The following commands allow you to perform programming operations from the remote terminal. With these commands you can check to make sure a device is blank, load data into RAM from master devices, and program blank devices with the master data. The commands are presented in the following order:

| Command Name | Command Format |
|---|---|
| Blank Check | B |
| Load Single | L [<start address> <end address> <device address>] |
| Program Single | P [<start address> <end address> <device address>] |
| Load Set | LS [<set size>] |
| Program Set | PS [<set size>] |
| Load Word | LW [<set size>] |
| Program Word | PW [<set size>] |
| Load Long Word | LL [<set size>] |
| Program Long Word | PL [<set size>] |

## Enabling/Disabling the Electrical Erase Function

The Electrical Erase function allows you to disable the ERASE function so that you may change some of the programming functions of an EEPROM device without erasing all of the programming information contained in that device. By disabling the ERASE function, for instance, you could change a location only and leave all of the other information contained in that device intact. To enable or disable the Electrical Erase function, perform the following procedure:

| Procedure | Example Key Sequence |
|---|---|
| 1. Type in the Electrical Erase Function command letters, EO, followed by a space. | EO |
| 2. If you want to enable the Electrical Erase feature, type a 1 and press carriage return. If you want to disable the Electrical Erase feature, type a 0 (zero) and press carriage return.<br><br>The programmer will display a message acknowledging that the Electrical Erase feature has either been enabled or disabled, depending upon whether you entered a 0 or a 1. | EO 1<CR><br>or<br>EO 0<CR> |

## Loading a Single Master

Use the Load Single command to load (or copy) data from a single master device into the programmer's data RAM for subsequent programming of blank devices, uploading or editing. You can load data into RAM starting at programmer RAM address 0, or you can specify a different block nf programmer RAM to be loaded with data. You can also select the device address that the first byte of data will be loaded from. Loading the master data into RAM is the first step in programming devices. The second step is to copy the data from RAM to a blank device using the Program Single command (which is explained following this subsection).

Command Format: L [<start address> <end address> <device address>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Make sure that the correct device type is selected. | |
| 2. Insert the master device into the leftmost socket, socket 1, and push the socket lever down to lock the device into place. | |
| 3. Type in the Load Single command letter, L. | `L` |
| 4. Type in the first address of RAM that you want data written to, or, to load RAM starting with the default programmer RAM address (0) and the default device address (0), go to step 7. (The space after "L" is optional.) | `L 0` |
| 5. Type a space and then type in the last address of RAM that you want data written into. If you entered a start address, you must enter an end address and a starting device address. | `L 0 1FFF` |

## Checking for Non-blank Devices

The Blank Check command allows you to check a device prior to programming it to make sure that it is blank.

Command Format: B

| Procedure | Example Key Sequence |
|---|---|
| 1. Make sure that the correct device type is selected. | |
| 2. Insert the device or devices to be checked in any socket or sockets and lock them into place by pushing the socket levers down. | |
| 3. Type in the Blank Check command letter, B, and press carriage return. | B<CR> |
| 4. Check the screen display for the letters that appear below the socket numbers of sockets containing devices. A "P" (for "pass") appears below the number of any socket containing a blank device and an "F" (for "fail") appears below the number of any socket containing a non-blank device. If the blank check is unsuccessful for any of the devices installed, the terminal will also display an error message. See the Error Messages section for an explanation of the error message. | |
| 5. Lift the socket lever(s) and remove the device(s). | |

## Programming Devices from a Single Master

Use the Program Single command to program from one to eight devices with the same block of master data loaded into the programmer's RAM. You can program the devices starting with the first byte of programmer RAM, or you can specify another range of programmer RAM to be programmed into the devices. You can also select the device address into which the first data byte will be written.

Command Format: P [<start address> <end address> <device address>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Make sure that the correct device type is selected. Programming a device with the wrong family and pinout codes selected could permanently damage the device. | |
| 2. Insert the blank device or devices in any of the eight sockets and lock each device into place. | |
| 3. Type in the Program Single command letter, P. | P |
| 4. Type in the first address of RAM that you want programmed into each device, or, to program each device starting with the default programmer RAM address (0) and the default device address (0), go to step 7. (The space after "P" is optional.) | P 0 |
| 5. Type a space and then type in the last address of RAM that you want programmed into each device. If you entered a start address, you must enter an end address. | P 0 1FFF |

| Procedure | Example Key Sequence |
|---|---|
| 6. Type a space and then type in the device address that you want the first byte of data to be loaded from. | L 0 1FFF 0 |
| 7. Press carriage return to execute the command.<br><br>When the load operation is complete, check the screen display for the letter that appears below device number 1. If the device was loaded successfully, a "P" will appear below the number 1 and the LED below socket 1 will light green. Also note the sumcheck total which is displayed below the device information. (The sumcheck is the hexadecimal total of all the data loaded.) The sumcheck should match the sumcheck of any devices programmed with this data.<br><br>If the device was not loaded successfully, an "F" will appear below device number 1 and the LED below socket 1 will light red. An error message will also be displayed on the terminal. See the Error Messages section for an explanation of the error message. | L 0 1FFF 0<CR><br>or<br>L<CR> |
| 8. Lift the socket lever and remove the master device. | |

## Loading a Set of Masters

Use the Load Set command to load data from a set of master devices into programmer RAM for subsequent programming into a set of blank devices, uploading, or editing. The data of each successive device is loaded into a successive block of programmer RAM. See the table following the procedure steps for the block of RAM assigned to each device loaded if the maximum set size is selected. After loading the set of master devices, you can use the Program Set command (described following this subsection) to program the data into a set or sets of blank devices.

Command Format: LS [<set size>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Make sure that the correct device type is selected. | |
| 2. Insert the set of masters into consecutive sockets, beginning at socket 1 (the leftmost socket), and push down the socket levers. (See the table following the procedure steps for the maximum number of devices that can be loaded with one Load Set operation and the programmer RAM blocks assigned to each device). The device data will be loaded into RAM sequentially, starting with address 0 of the leftmost device and ending with the last address of the rightmost device of the specified set size. | |
| 3. Type in the Load Set command letters, LS. | LS |

| Procedure | Example Key Sequence |
|---|---|
| 6. Type a space and then type in the address of the device memory location that you want the first byte of data to be written to (destination address). The default destination address is 0. | `P 0 1FFF 0` |
| 7. Press carriage return to execute the command.<br><br>When the program operation is complete, check the screen display for the letters that appear below the socket numbers of sockets containing devices. If a device was programmed successfully, a "P" will appear below the number of the socket containing the device and the LED below the socket will light green. If a device was not programmed successfully, an "F" will appear below the number of the socket containing the device and the LED below the socket will light red. If any of the installed devices failed to program correctly, an error message will also be displayed on the terminal screen. If all of the installed devices were programmed successfully, note the sumcheck total which is displayed below the device information. The sumcheck should match the sumcheck of the data in the programmer's RAM. | `P 0 1FFF 0<CR>`<br>or<br>`P<CR>` |
| 8. Lift the socket lever(s) and remove the device(s). | |

Programmer RAM blocks assigned to a set of devices being loaded if the programmer has 512K of RAM and the maximum set size is selected:

| Socket Number | Device Size (no. of bits) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 16K | 32K | 64K | 128K | 256K | 512K | 1M |
| 1 | 000 | 000 | 0000 | 0000 | 0000 | 0000 | 00000 |
| | 7FF | FFF | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF |
| 2 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 |
| | FFF | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF |
| 3 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 | 40000 |
| | 17FF | 2FFF | 5FFF | BFFF | 17FFF | 2FFFF | 5FFFF |
| 4 | 1800 | 3000 | 6000 | C000 | 18000 | 30000 | 60000 |
| | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF | 7FFFF |
| 5 | 2000 | 4000 | 8000 | 10000 | 20000 | 40000 | |
| | 27FF | 4FFF | 9FFF | 13FFF | 27FFF | 4FFFF | |
| 6 | 2800 | 5000 | A000 | 14000 | 28000 | 50000 | |
| | 2FFF | 5FFF | BFFF | 17FFF | 2FFFF | 5FFFF | |
| 7 | 3000 | 6000 | C000 | 18000 | 30000 | 60000 | |
| | 37FF | 6FFF | DFFF | 1BFFF | 37FFF | 6FFFF | |
| 8 | 3800 | 7000 | E000 | 1C000 | 38000 | 70000 | |
| | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF | 7FFFF | |

Programmer RAM blocks assigned to a set of devices being loaded if the programmer has 512K of RAM and the maximum set size is selected:

| Socket Number | Device Size (no. of bits) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 16K | 32K | 64K | 128K | 256K | 512K | 1M |
| 1 | 000 | 000 | 0000 | 0000 | 0000 | 0000 | 00000 |
| | 7FF | FFF | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF |
| 2 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 |
| | FFF | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF |
| 3 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 | 40000 |
| | 17FF | 2FFF | 5FFF | BFFF | 17FFF | 2FFFF | 5FFFF |
| 4 | 1800 | 3000 | 6000 | C000 | 18000 | 30000 | 60000 |
| | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF | 7FFFF |
| 5 | 2000 | 4000 | 8000 | 10000 | 20000 | 40000 | |
| | 27FF | 4FFF | 9FFF | 13FFF | 27FFF | 4FFFF | |
| 6 | 2800 | 5000 | A000 | 14000 | 28000 | 50000 | |
| | 2FFF | 5FFF | BFFF | 17FFF | 2FFFF | 5FFFF | |
| 7 | 3000 | 6000 | C000 | 18000 | 30000 | 60000 | |
| | 37FF | 6FFF | DFFF | 1BFFF | 37FFF | 6FFFF | |
| 8 | 3800 | 7000 | E000 | 1C000 | 38000 | 70000 | |
| | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF | 7FFFF | |

| Procedure | Example Key Sequence |
|---|---|
| 4. Type in the number of masters in the set to be loaded and press carriage return, or, to load the default number of masters, press carriage return without entering a number of devices. The power-up default number of masters is 1. (The space after "LS" is optional.)<br><br>If a socket to the left of the last master is left empty, "FF" will be loaded into the RAM block assigned to that socket. If devices are installed to the right of the last master of the set size specified, the extra devices will not be loaded. | LS 4<CR><br>or<br>LS<CR> |
| 5. After the load operation is complete, check the screen display for the letters that appear below the numbers of sockets containing devices. A "P" will appear below the number of any socket containing a device that was loaded successfully and an "F" will appear below the number of any socket containing a device that was not loaded successfully. If any of the devices were not loaded successfully, an error message will also be displayed on the terminal screen. (See the Error Messages section for an explanation of the error message.) If the entire set of devices was loaded successfully, note the sumcheck total which appears below the device information. The sumcheck total should match the sumcheck of any set of devices programmed with this data. | |
| 6. Lift the socket levers and remove the master devices. | |

## Programming Sets of Devices

Use the Program Set command to program one or more sets of devices with the contents of the programmer's RAM. This command fills the devices of a set with successive blocks of programmer RAM, starting with RAM address 0 and device address 0. The programmer fills the leftmost device with the first block of RAM and the next device with the next (consecutive) block of RAM, and so on, up to the set size. If additional devices are installed beyond the set size, the programmer begins producing a duplicate of the first set, filling the first device of the second set with the first block of programmer RAM (starting at address 0) and the second device of the second set with the next (consecutive) block of programmer RAM, and so on. See the examples following the procedure steps for a demonstration of which block of RAM will be programmed into which device. You can also use this command to merge the master device data into larger devices. To merge the data into larger devices, select the device type of the larger blank devices before executing the Program Set command and select the appropriate set size.

Command Format: PS [<set size>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Make sure that the correct device type is selected. Programming a device with the wrong family and pinout codes selected could permanently damage the device. | |
| 2. Insert the set or sets of blank devices into consecutive sockets, starting with socket 1, and lock them into place. | |
| 3. Type in the Program Set command letters, PS. | PS |

## Loading Word-Wide Masters

Use the Load Word command to load pairs of 8-bit-wide devices in 16-bit-wide word format. This command reads the first address of the first device and then the first address of the second device into consecutive RAM locations. The addresses are loaded in pairs, as described, consecutively from the first pair to the last pair (see illustration). You can load more than one pair of 8-bit-wide devices (with the data for each successive pair being loaded into successive RAM blocks), as long as the total size of all the devices does not exceed the size of the programmer's RAM. For 16-bit-wide devices, the Load Word command functions exactly like the Load Single command.

Command Format: LW [<set size>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Make sure that the correct device type is selected. | |
| 2. Insert the pair or pairs of 8-bit-wide devices into the sockets, starting with socket 1, and lock them into place. Place the devices of each pair in adjacent sockets, with the device containing the high-order data to the right of the device containing the low-order data for the pair. | |
| 3. Type in the Load Word command letters, LW. | LW |

*Example 1*

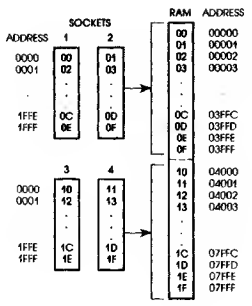
Programmer RAM blocks programmed into sets of maximum set size if the programmer has 512K of memory:

| Socket Number | Device Size (no. of bits) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 16K | 32K | 64K | 128K | 256K | 512K | 1M |
| 1 | 000 | 000 | 0000 | 0000 | 0000 | 0000 | 00000 |
| | 7FF | FFF | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF |
| 2 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 |
| | FFF | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF |
| 3 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 | 40000 |
| | 17FF | 2FFF | 5FFF | BFFF | 17FFF | 2FFFF | 5FFFF |
| 4 | 1800 | 3000 | 6000 | C000 | 18000 | 30000 | 60000 |
| | 1FFF | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF | 7FFFF |
| 5 | 2000 | 4000 | 8000 | 10000 | 20000 | 40000 | 00000 |
| | 27FF | 4FFF | 9FFF | 13FFF | 27FFF | 4FFFF | 1FFFF |
| 6 | 2800 | 5000 | A000 | 14000 | 28000 | 50000 | 20000 |
| | 2FFF | 5FFF | BFFF | 17FFF | 2FFFF | 5FFFF | 3FFFF |
| 7 | 3000 | 6000 | C000 | 18000 | 30000 | 60000 | 40000 |
| | 37FF | 6FFF | DFFF | 1BFFF | 37FFF | 6FFFF | 5FFFF |
| 8 | 3800 | 7000 | E000 | 1C000 | 38000 | 70000 | 60000 |
| | 3FFF | 7FFF | FFFF | 1FFFF | 3FFFF | 7FFFF | 7FFFF |

*Example 2*

RAM blocks programmed into sets of four 64K bit devices:

| Socket Number | RAM Block |
|---|---|
| 1 | 0000 |
| | 1FFF |
| 2 | 2000 |
| | 3FFF |
| 3 | 4000 |
| | 5FFF |
| 4 | 6000 |
| | 7FFF |
| 5 | 0000 |
| | 1FFF |
| 6 | 2000 |
| | 3FFF |
| 7 | 4000 |
| | 5FFF |
| 8 | 6000 |
| | 7FFF |




*NOTE*

*The above illustration represents the loading of two pairs of 64K devices.*

| Procedure | Example Key Sequence |
|---|---|
| 4. Type in the number of pairs to be loaded and press carriage return, or, to load the default number of pairs, press carriage return without entering a number. The power-up default number of pairs is 1. (The space after "LW" is optional.) Any devices installed to the right of the last master of the set specified will not be loaded. | `LW 4<CR>`<br>or<br>`LW<CR>` |
| 5. After the load operation is complete, check the screen display for the letters that appear below the numbers of the sockets containing devices. A "P" will appear below the number of any socket containing a device that was loaded successfully and an "F" will appear below the number of any socket containing a device that was not loaded successfully. If any of the devices in the set were not loaded successfully, an error message will also be displayed on the terminal screen. If all of the pairs in the set were loaded successfully, note the sumcheck total which appears below the device information. The sumcheck should match the sumcheck of any set(s) of devices programmed with this data. | |
| 6. Lift the socket levers and remove the devices. | |

| Procedure | Example Key Sequence |
|---|---|
| 4. Type in the total number of device pairs to be programmed, set size, and press carriage return, or, to program using the default set size, press carriage return without entering a number of devices. (The space after "PW" is optional.) The power-up default set size is 1. Any devices installed in the sockets beyond the set size will be programmed as duplicates of the original set. For example, if you install 6 devices and specify a set size of 2, the device in socket 5 will be programmed as a duplicate of the device in socket 1 and the device in socket 6 will be programmed as a duplicate of the device in socket 2. | `PW 2<CR>`<br>or<br>`PW<CR>` |
| 5. After the program operation is complete, check the screen display for the letters that appear below the numbers of sockets containing devices. A "P" will appear below the number of any socket containing a device that was programmed successfully and an "F" will appear below the number of any socket containing a device that was not programmed successfully. If any of the installed devices failed to program correctly, an error message will be displayed on the screen. If all of the installed devices were programmed successfully, note the sumcheck total which appears below the device information. The sumcheck should match the sumcheck of the data in the programmer's RAM. | |
| 6. Lift the socket levers and remove the programmed devices. | |

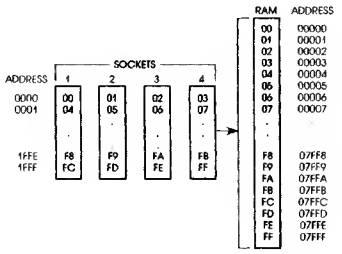
## Programming Devices Using a Word-Wide Format

Use the Program Word command to program 8-bit-wide devices with data loaded into the programmer's RAM in 16-bit-wide word format. The programmer programs the low bytes (even-addressed bytes) into the first device of the pair and the high bytes (odd-addressed bytes) into the second device of the pair. You can program up to four duplicate pairs of word-wide devices using one Program Word command, or up to four pairs containing different data, provided the programmer memory has been loaded with data for all of the pairs. The programmer determines which RAM data will be written into the additional pairs of devices depending upon the set size you specify for the program operation. For example, if you specify a set size (number of pairs) of 2, the devices in sockets 1 and 2 will be programmed as a pair with the first block of data and the devices in sockets 3 and 4 will be programmed as a different pair with the next block of data. However, if you install devices in socket 5, 6, 7 and 8 as well, these devices will be programmed as duplicates of the first two pairs of devices. You can also use this command to merge the data from pairs of smaller master devices into one or more pairs of larger devices by specifying the device type of the larger devices and specifying the appropriate set size.

Command Format: PW [<set size>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Make sure that the correct device type is selected. Programming a device with the wrong family and pinout codes selected could permanently damage the device. | |
| 2. Insert the device pairs into the sockets and lock them into place. The devices in sockets 1 and 2 will be programmed as a 16-bit-wide word pair. Each successive pair of devices will be programmed as a separate 16-bit-wide pair. | |
| 3. Type in the Program Word command letters, PW. | PW |

| Procedure | Example Key Sequence |
|---|---|
| 5. After the load operation is complete, check the screen display for the letters that appear below the numbers of the sockets containing devices. A "P" will appear below the number of any socket containing a device that was loaded successfully and an "F" will appear below the number of any socket containing a device that was not loaded successfully. If any of the devices in the set were not loaded successfully, an error message will also be displayed on the terminal. If all of the devices of the set were loaded successfully, note the sumcheck total which appears below the device information. The sumcheck should match the sumcheck of any set(s) of devices programmed with this data. | |
| 6. Lift the socket levers and remove the master devices. | |




*NOTE*

*The illustration represents the loading of a set of 64K bit devices.*

## Loading Long-Word-Wide Masters

Use the Load Long Word command to load sets of four 8-bit-wide devices in 32-bit-wide word format. See the illustration following the procedure steps for the order that device data is loaded into RAM. You can load two unique long-word-wide sets of 8-bit-wide devices, as long as the total size of all of the devices does not exceed the programmer's RAM.

Command Format: LL [<set size>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Make sure that the correct device type is selected. | |
| 2. Insert the set(s) of 8-bit-wide devices into the sockets, starting with socket 1, and lock them into place. Place the devices of a set in order (from left to right) in adjacent sockets, with the lowest-order device for the set furthest to the left. | |
| 3. Type in the Load Long Word command letters, LL. | LL |
| 4. Type in the number of sets to be loaded (1 or 2) and press carriage return, or, to load the default number of sets, press carriage return without entering a number of masters. (The space after "LL" is optional.) The power-up default set size is 1. Any devices installed to the right of the last master of the set size specified will not be loaded. | LL 8<CR><br>or<br>LL<CR> |

| Procedure | Example Key Sequence |
|---|---|
| 4. Type in the number of sets of devices to be programmed, set size (1 or 2), and press carriage return, or, to program using the default set size, press carriage return without entering a number of devices. (The space after "PL" is optional.) The power-up default set size is 1. | PL 2<CR><br>or<br>PL<CR> |
| 5. After the program operation is complete, check the screen display for the letters that appear below the numbers of sockets containing devices. A "P" will appear below the number of any socket containing a device that was programmed successfully and an "F" will appear below the number of any socket containing a device that was not programmed successfully. If any of the installed devices failed to program correctly, an error message will be displayed on the screen. If all of the installed devices programmed successfully, note the sumcheck total which appears below the device information. The sumcheck should match the sumcheck of the data in the programmer's RAM. | |
| 6. Lift the socket levers and remove the programmed devices. | |

## Programming Devices Using a Long-Word-Wide Format

Use the Program Long Word command to program 8-bit-wide devices with data loaded into the programmer's RAM in 32-bit-wide word format. This command treats four 8-bit-wide devices as a single 32-bit-wide device. The programmer programs address 0 of each of the four devices of a set with the first four successive bytes of memory, and then programs address 1 of each of the four devices of the set with the next four successive bytes of memory, and so on (see the previous illustration). You can program two unique sets of long-word-wide devices with one Program Long Word command, provided the data for both sets has been loaded into programmer RAM. The second set of devices will be programmed with the block of memory following the last byte of memory programmed into the first set of devices. You can also program two sets of identical long-word-wide devices with one Program Long Word command by specifying a set size of 2 and inserting 8 blank devices. (Partial sets can also be programmed by inserting a partial set of blank devices.)

Command Format: PL [<set size>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Make sure that the correct device type is selected. Programming a device with the wrong family and pinout codes selected could permanently damage the device. | |
| 2. Insert the set(s) of devices into the sockets and lock them into place. The devices in sockets 1, 2, 3, and 4 will be programmed as one 32-bit-wide word set and devices in sockets 5, 6, 7, and 8 will be programmed as a second 32-bit-wide word set. The second set will be programmed as a duplicate set of the first set if you specify a set size of 1, or as a unique set if you specify a set size of 2. | |
| 3. Type in the Program Long Word command letters, PL. | PL |

## Verifying Devices Against a Single Master

The Verify Single command allows you to verify a single device or up to eight identical devices simultaneously against the master data loaded into the programmer's memory. You can verify a device against RAM data beginning at programmer RAM address 0, or you can select a different block of RAM to verify the device against. You can also select the address of the device memory location where the verify operation will begin.

Command Format: V [<Vcc> <start address> <end address> <device address>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Load the master data into the programmer's RAM using either a Load Single operation or a download operation. | |
| 2. Make sure that the correct device type is selected. | |
| 3. Insert the device or devices to be verified into any of the sockets. Each device will be verified against the same block of RAM. | |
| 4. Type in the Verify Single command letter, V. | V |

## VERIFYING PROGRAMMED DEVICES

Use the following commands to verify device data against data loaded into the programmer's RAM. Each device is automatically verified after it is programmed; however, if you wish to verify devices at another time, you can use these commands. Before you use any of the following verify commands, you must load the programmer's RAM with the data which the devices are to be verified against using either a load or a download command. The verify commands are presented in the following order:

| Command Name | Command Format |
|---|---|
| Verify Single | V [<Vcc>] [<start address> <end address> <device address>] |
| Verify Set | VS [<Vcc>] [<set size>] |
| Verify Word | VW [<Vcc>] [<set size>] |
| Verify Long Word | VL [<Vcc>] [<set size>] |

9. Press carriage return to execute the verify command.

```
V L 0 1FFF 0<CR>
      or
V L<CR>
```

After the verify operation is complete, check the screen display for the letters that appear below the numbers of sockets containing devices. A "P" will appear below the number of any socket containing a device that was verified successfully and an "F" will appear below the number of any socket containing a device that was not verified successfully. If any of the devices failed to verify, an error message will also be displayed on the screen. If all of the installed devices verified, note the sumcheck total which appears below the device information. The sumcheck should match the sumcheck produced when the device was programmed.

10. Lift the socket lever(s) and remove the verified device(s).

| Procedure | Example Key Sequence |
|---|---|
| 5. Type a space and then type the letter (L or H) that corresponds to the proper Vcc high and low settings shown in the following table (in this case, the space following the command letter V is required); or, to verify the installed device with the voltage settings recommended bythe manufacturer, do not type anything for this parameter.<br>Selection High Voltage Low Voltage<br>L 5.2 V 4.8 V<br>H 5.5 V 4.5 V | V L |
| 6. Type in the first address of RAM that you want the device to be verified against, or, to verify the device against the entire block of RAM data assigned to that device, go to step 9. | V L 0 |
| 7. Type a space and then type in the last address of RAM that you want the device to be verified against. If you entered a start address, you must enter an end address and a starting device address. | V L 0 1FFF |
| 8. Type a space and then type in the first address of device memory that you want to be verified against the selected block of RAM. | V L 0 1FFF 0 |

| Procedure | Example Key Sequence |
|---|---|
| 5. Type in the letter that corresponds to the proper Vcc high and low settings shown in the following table, or, to verify the installed sets with the voltage settings recommended by the manufacturer, do not type anything for this parameter. (The space after "VS" is optional.)<br>Selection High Voltage Low Voltage<br>L 5.2 V 4.8 V<br>H 5.5 V 4.5 V | `VS L` |
| 6. Type a space, type in the number of devices in the set (set size) and press carriage return, or, to verify the set(s) using the default set size, press carriage return without entering a number of devices. The power-up default set size is 1. | `VS L 4<CR>`<br>or<br>`VS L<CR>` |
| 7. After the verify operation is complete, check the screen display for the letters that appear below the numbers of sockets containing devices. If any device failed to verify, an "F" appears below the socket number of the socket containing that device and an error message is displayed on the screen. If all of the installed devices verified, note the sumcheck total which appears below the device information. The sumcheck should match the sumcheck of the set data loaded into RAM. | |
| 8. Lift the socket levers and remove the verified devices. | |

## Verifying Sets of Devices

The Verify Set command allows you to verify sets of devices (each containing different data) against the master set data stored in the programmer's RAM. You can also verify multiple copies of the same set with one Verify Set command. (See Programming a Set of Devices for the table showing which block of RAM is assigned to which socket number.)

Command Format: VS [<Vcc>] [<set size>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Load the master set data into the programmer's RAM using either a Load Set command or a download operation. | |
| 2. Make sure that the correct device type is selected. | |
| 3. Insert the set(s) of devices in the sockets so that the device data will be verified against the same block of data that it was originally programmed with. Any devices inserted beyond the set size specified, will be verified as duplicates of the leftmost set. (See Programming a Set of Devices for the table showing which block of RAM is assigned to which socket number.) | |
| 4. Type in the Verify Set command letters, VS. | VS |

| Procedure | Example Key Sequence |
|---|---|
| 5. Type in the letter that corresponds to the proper Vcc high and low settings shown in the following table, or, to verify the installed device with the voltage settings recommended by the manufacturer, do not type anything for this parameter. (The space after "VW" is optional.)<br>Selection High Voltage Low Voltage<br>L 5.2 V 4.8 V<br>H 5.5 V 4.5 V | VW L |
| 6. Type a space, type in the number of device pairs to be verified, set size, and press carriage return, or, to verify using the default set size, press carriage return without entering a number of devices. The power-up default set size is 1. | VW L 2<CR><br>or<br>VW L<CR> |
| 7. After the verify operation is complete, check the screen display for the letters that appear below the numbers of sockets containing devices. If any device failed to verify, an "F" appears below the socket number of the socket containing that device and an error message is displayed on the screen. If all of the installed devices verified, note the sumcheck total which appears below the device information. The sumcheck should match the sumcheck of the set data loaded into RAM. | |
| 8. Lift the socket levers and remove the verified devices. | |

## Verifying Word-Wide Device Pairs

Use the Verify Word command to verify data stored in 8-bit-wide device pairs against data that is loaded into RAM in 16-bit-wide word format. This command compares address 0 of the first device with the first byte of RAM, address 0 of the second device with the second byte of RAM, address 1 of the first device with the third byte of RAM, and so on. Up to four different pairs of devices can be compared, provided the master data for all pairs to be verified is loaded into the programmer's RAM. Also, multiple copies of each pair can be verified with one command.

Command Format: VW [<Vcc>] [<set size>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Load the master set data into the programmer's RAM using either a Load Word command or a download operation. | |
| 2. Make sure that the correct device type is selected. | |
| 3. Insert the pair or pairs of devices (up to 4) in the sockets, starting withsocket 1. Place the devices in the sockets so that the devices of a pairare adjacent, with the high-order device to the right of the low-order device for the pair. | |
| 4. Type in the Verify Word command letters, VW. | VW |

| Procedure | Example Key Sequence |
|---|---|
| 5. Type in the letter that corresponds to the proper Vcc high and low settings shown in the following table, or, to verify the installed devices with the voltage settings recommended by the manufacturer, do not type anything for this parameter. (The space after "VL" is optional.)<br><br>Selection High Voltage Low Voltage<br><br>L 5.2 V 4.8 V<br>H 5.5 V 4.5 V | VL L |
| 6. Type a space, type in the number of sets to be verified, set size, (1 or 2) and press carriage return, or, to verify using the default set size, press carriage return without entering a number of devices. The power-up default set size is 1. | VL L 2<CR><br>or<br>VL L<CR> |
| 7. After the verify operation is complete, check the screen display for the letters that appear below the numbers of sockets containing devices. If any device failed to verify, an "F" appears below the socket number of the socket containing that device and an error message is displayed on the screen. If all of the installed devices verified, note the sumcheck total which appears below the device information. The sumcheck should match the sumcheck of the set data loaded into RAM. | |
| 8. Lift the socket levers and remove the verified devices. | |

## Verifying Long-Word-Wide Devices

Use the Verify Long Word command to verify data stored in 8-bit-wide device sets against data that is loaded into RAM in 32-bit-wide word format. This command compares four 8-bit-wide devices to 32-bit-wide data in the programmer's RAM. (See Loading Long-Word-Wide Masters for information on which device number corresponds to which bytes of RAM data.) Two unique sets of devices can be verified simultaneously with one Verify Long Word command, provided that the data for both sets is loaded into the programmer's RAM, or, two copies of the same set can be verified with one Verify Long Word command.

Command Format: VL [<Vcc>] [<set size>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Load the master data into the programmer's RAM using either a Load Long Word command or a download operation. | |
| 2. Make sure that the correct device type is selected. | |
| 3. Insert the set or sets of devices into the sockets, starting with socket 1. Place the devices of each set in order, with the device containing the lowest-order bits furthest to the left for each set. Make sure that the RAM data that the device was programmed with is the same RAM data that the device will be verified against. | |
| 4. Type in the Verify Long Word command letters, VL. | VL |

## Displaying Memory (Byte or Word)

Use the Memory Display command to display the contents of RAM on the terminal screen. The RAM addresses are arranged on the screen in a table format, with each row containing 16 bytes of consecutive data (see example) or 8 words of consecutive data. To determine the address of a byte of RAM, add the 6-character RAM address shown at the very left of the line containing the data to the 2-character address shown at the top of the column containing the data. The ASCII-CODE column on the far right contains the ASCII equivalent of the hexadecimal data on each line.

Command Format for Byte Display: M <start address> <end address>
Command Format for Word Display: MW <start address> <end address>

| Procedure | Example Key Sequence |
|---|---|
| 1. Type the Memory Display command letter, M (or MW). | M |
| 2. Type in the address of the first byte of RAM that you want to display on the screen. | M 0 |
| 3. Type a space, type the address of the last byte of RAM that you want to display on the screen and press carriage return. | M 0 1F<CR> |

4. Type CTRL-S to halt the display and CTRL-Q to resume the display.

## EDITING MEMORY

The following commands allow you to edit data that has been loaded into the 288A Multi Programmer's RAM. With these commands, you can display the data, edit individual addresses, fill a segment of memory with data, insert new data, delete data, copy data, and search memory for up to 8 bytes of data. The commands are presented in the following order:

| Command Name | Command Format |
|---|---|
| Memory Display | M <start address> <end address> |
| Memory Modify | M <start address> |
| Memory Fill | M <start address> <end address> <data1> [<data2...data8>] |
| Memory Display Word | MW <start address> <end address> |
| Memory Modify Word | MW <start address> |
| Memory Fill Word | MW <start address> <end address> <data1> [<data2...data8>] |
| Insert | I <start address> <data1> [<data2...data8>] |
| Delete | D <start address> [<end address>] |
| Transfer (Copy) | T <start address> <end address> <destination address> |
| Memory Search | MS <start address> <end address> <data1> [<data2...data8>] |
| Byte Swap | BS <start address> <end address> |

You may also want to use the Sumcheck (Total) command to calculate a sumcheck of the data stored in the programmer's memory after you have finished editing and are ready to program devices. The Sumcheck (Total) command is explained following this subsection.

## Modifying Single Memory Locations (Byte or Word)

Use the Memory Modify command to display and edit the contents of RAM locations one-by-one. The Memory Modify command causes the programmer to enter an editing mode which allows you to display the contents of RAM addresses byte-by-byte or word-by-word and type in new data for each address displayed. To exit Memory Modify mode and return to the command prompt (>), press <ESC>.

Byte Modify
Command Format: M <start address><CR>
<new data><CR>
.
.
.
<ESC>

Word Modify
Command Format: MW <start address><CR>
<new data><CR>
.
.
.
<ESC>

| Procedure | Example Key Sequence |
|---|---|
| 1. Type the Memory Modify command letter, M (or MW). | M |
| 2. Type in the address of the first memory location you want to modify and press carriage return. | M 6<CR> |
| 3. After the RAM address and its contents have been displayed, type in the new data and press carriage return to accept it. The new data will appear to the right of the old data. If you do not want to modify the data at the address displayed, but want to display the next RAM address for modification, press carriage return without entering new data. | E5<CR><br>or<br><CR> |

4. To modify the next memory location, return to step 3.

### Example

Display the contents of RAM addresses 0 through 1F:

```
>M 0 1F<CR>
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000000 32 1A 5B 47 43 52 03 4B CB 20 54 33 4A 11 20 52  2.[GCR.K. T3J. R
000010 52 20 11 4A 33 54 20 CB 4B 03 AA 52 43 47 4B 1A  R.J3T .K..RCGK..
```

*NOTE*

*To determine the memory address of a data byte, add the 2-character column heading of the data byte column to the row heading (6-character hexadecimal number at the far left of each line).*

*A period under the ASCII-CODE heading represents hexadecimal data without a printable ASCII equivalent.*

## Filling a Segment of Memory (Byte or Word)

Use the Memory Fill command to fill (replace) a segment of RAM with up to eight bytes or words of new data. The original data in these RAM locations are overwritten with the new data.

Byte Modify
Command Format: M <start address> <end address> <data1> [<data2>...<data8>]

Word Modify
Command Format: MW <start address> <end address> <data1> [<data2>...<data8>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Type the Memory Fill command letter, M (or MW). | M |
| 2. Type in the address of the first byte of RAM you want to fill with new data. | M 20 |
| 3. Type a space and then type in the address of the last byte of RAM you want to fill with new data. | M 20 2F |
| 4. Type a space, type in the data to be input into the specified memory range and press carriage return. Separate each hexadecimal byte with a space. You can insert up to eight hexadecimal bytes (or eight ASCII characters) with one Memory Fill command. If you input more data than can fit in the range specified, the excess data is ignored. | M 20 2F 11 22<CR> |

| Procedure | Example Key Sequence |
|---|---|

5. To exit Memory Modify mode and return to the command prompt (>), press <ESC>. If you press <ESC> after typing in new data but before pressing carriage return to accept the new data, the new data for the last address displayed will be ignored.

### Example

Enter Memory Modify mode starting at address 6 and type in new data for addresses 6, 7, and 8:

```
>M 6<CR>
000006  F1E5<CR>
000007  FF21<CR>
000008  0021<CR>
000009  43<ESC>
```

## Inserting New Data

Use the Insert command to insert new data into RAM. The original data is not overwritten by this command, as it is with the Memory Modify and the Memory Fill commands, but is shifted up (to a higher address) to accommodate the new data. Data at the end of memory is lost.

Command Format: I <start address> <data1> [<data2>...<data8>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Type the Insert command letter, I. | I |
| 2. Type in the starting address where you want the new data to be inserted. | I 5 |
| 3. Type a space, type in the new data and then press carriage return. Separate each hexadecimal byte with a space. You can insert up to eight hexadecimal bytes (or eight ASCII characters) with one Insert command. | I 5 11 11 11<CR> |

### Example

Display memory contents before fill:

```
>M 20 3F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000020  FF FF FF FF EE EE EE EE DD DD DD DD CC CC CC CC  ................
000030  BB BB BB BB AA AA AA AA 99 99 99 99 88 88 88 88  ................
```

Fill memory:

```
>M 20 2F 11 22<CR>
```

Display memory contents after fill:

```
>M 20 3F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000020  11 22 11 22 11 22 11 22 11 22 11 22 11 22 11 22  ."."."."."."."."
000030  BB BB BB BB AA AA AA AA 99 99 99 99 88 88 88 88  ................
```

## Deleting Data

Use the Delete command to delete data bytes from memory. Data following the deleted addresses are shifted up (to a lower address) to fill in the deleted addresses. "FF" is filled in the vacated addresses at the end of RAM.

Command Format: D <start address> [<end address>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Type the Delete command letter, D. | D |
| 2. Type in the address where you want to begin deleting data. | D 4 |
| 3. To delete a single byte of data, press carriage return without entering an end address. To delete a range of data, type a space, type the address of the last memory byte that you want to delete and then press carriage return. | D 4<CR><br>or<br>D 4 7<CR> |

### Example

Display memory contents before insertion of new data:

```
>M 0 1F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000000  FF FF FF FF EE EE EE EE DD DD DD DD CC CC CC CC  ................
000010  BB BB BB BB AA AA AA AA 99 99 99 99 88 88 88 88  ................
```

Insert new data:

```
>I 5 11 11 11 11<CR>
```

Display memory contents after insertion of new data:

```
>M 0 1F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000000  FF FF FF FF EE 11 11 11 11 EE EE EE DD DD DD DD  ................
000010  CC CC CC CC BB BB BB BB AA AA AA AA 99 99 99 99  ................
```

## Transferring (Copying) Memory

Use the Transfer command to copy a block of RAM data from one RAM location to another. This command copies the block specified by the start address and end address to the destination address. The original data starting at the destination address is overwritten by the copied block.

Command Format: T <start address> <end address> <destination address>

| Procedure | Example Key Sequence |
|---|---|
| 1. Type the Transfer command letter, T. | T |
| 2. Type in the address of the first byte of RAM to be copied. | T 0 |
| 3. Type a space and then type in the address of the last byte of RAM to be copied. | T 0 F |
| 4. Type a space and then type in the address that you want the first byte of the specified data range to be copied to, or the destination address, and press carriage return. The data will be copied into consecutive RAM addresses, starting with the destination address. | T 0 F 25<CR> |

### Example

Display memory contents before deletion:

```
>M 0 1F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000000  66 66 66 66 11 11 11 11 77 77 77 77 77 77 77 77  FFFF....WWWWWWWW
000010  88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88  ................
```

Delete data:

```
>D 4 7<CR>
```

Display memory contents after deletion:

```
>M 0 1F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000000  66 66 66 66 77 77 77 77 77 77 77 77 88 88 88 88  FFFFWWWWWWWW....
000010  88 88 88 88 88 88 88 88 88 88 88 88 FF FF FF FF  ................
```

## Searching Memory

Use the Memory Search command to find a block of hexadecimal data up to eight bytes long.

Command Format: MS <start address> <end address> <data1> [<data2>...<data8>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Type the Memory Search command letters, MS. | MS |
| 2. Type in the address of the memory location where you want the search to begin. | MS 0 |
| 3. Type a space and then type in the memory address where you want the search to end. | MS 0 3F |
| 4. Type a space and then type in the data that you want to search for and press carriage return. Separate each hexadecimal byte with aspace. You can type in up to eight hexadecimal bytes.<br><br>If the data is found, the terminal will display MEMORY MATCH AT ADDRESS HHHH, where "HHHH" is the address of the first byte of the search string that was typed in.<br><br>If the data is not found, the terminal will display MEMORY SEARCH FAILURE. | MS 0 3F 12 34<CR> |

### Example

Display memory contents before transfer:

```
>M 0 3F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000000  99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99  ................
000010  77 77 77 77 77 77 77 77 77 77 77 77 77 77 77 77  WWWWWWWWWWWWWWWW
000020  66 66 66 66 66 44 44 44 44 44 44 44 44 44 44 44  FFFFFDDDDDDDDDDD
000030  44 44 44 44 44 11 11 11 11 11 11 11 11 11 11 11  DDDDD...........
```

Transfer data:

```
>T 0 F 25<CR>
```

Display memory contents after transfer:

```
M 0 3F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000000  99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99  ................
000010  77 77 77 77 77 77 77 77 77 77 77 77 77 77 77 77  WWWWWWWWWWWWWWWW
000020  66 66 66 66 66 99 99 99 99 99 99 99 99 99 99 99  FFFFF...........
000030  99 99 99 99 99 11 11 11 11 11 11 11 11 11 11 11  ................
```

## Byte Swapping a Segment of Memory

Use the Byte Swap command to swap adjacent bytes of data throughout a segment of memory. The segment of memory must have an even number of bytes.

Command Format: BS <start address> <end address>

| Procedure | Example Key Sequence |
|---|---|
| 1. Type the Byte Swap command letters, BS. | BS |
| 2. Type in the address of the first byte of RAM to be swapped. | BS 0 |
| 3. Type a space and then type in the address of the last byte of RAM to be swapped (must be an even number of bytes) and press ENTER. The data will be swapped in adjacent bytes of RAM throughout the memory segment. | BS 0 1F<CR> |

### Example

Search memory block 0 through 3F for the hexadecimal data 12 34:

```
>MS 0 3F 12 34<CR>
MEMORY MATCH AT ADDRESS 000038
```

Display memory block searched:

```
>M 0 3F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   ASCII-CODE
000000  77 77 77 77 77 77 77 77 77 77 77 77 77 77 77 77   WWWWWWWWWWWWWWWW
000010  77 77 77 77 77 77 77 77 77 77 77 77 77 77 77 77   WWWWWWWWWWWWWWWW
000020  77 77 77 77 77 77 77 77 77 77 77 77 77 77 77 77   WWWWWWWWWWWWWWWW
000030  77 77 77 77 77 77 77 77 12 34 77 77 77 77 77 77   WWWWWWWW.4WWWWWW
```

# SUMCHECKING DATA

Use the following commands to cause the programmer to sumcheck (total) or checksum (EXOR) the data stored in the programmer's RAM. These commands are useful for obtaining a new sumcheck of any range of RAM, or verifying that the data stored in RAM is the same as data programmed into a device or data downloaded through the serial port. The 4-byte sumcheck may also be stored in RAM. This feature allows you to program the sumcheck into a device so that software which calculates a sumcheck during a self test can verify its own sumcheck against the sumcheck stored in the device.

## Performing a Sumcheck (Total)

Use the Sumcheck (Total) command to calculate a 4-byte hexadecimal summation of data stored in the programmer's RAM. This sumcheck can be used to verify that data was downloaded to the programmer correctly, to sumcheck only a portion of memory, or to produce a current sumcheck after you have edited memory. A sumcheck is automatically performed after most programming operations, so you should not need to use this command when programming or verifying devices using master devices.

Command Format: CT <start address> <end address> [<destination address>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Type in the Sumcheck (Total) command letters, CT. | CT |
| 2. Type in the address of the first byte of RAM to be checked by the sumcheck operation. | CT 0 |
| 3. Type a space and then type in the address of the last byte of RAM to be checked. | CT 0 3FFF |

### Example

Display memory contents before swap:

```
>M 0 1F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000000  01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16  ................
000010  17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32  ... !"#$%&'()012
```

Swap bytes:

```
>BS 0 1F<CR>
```

Display memory contents after swap:

```
>M 0 1F<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ASCII-CODE
000000  02 01 04 03 06 05 08 07 10 09 12 11 14 13 16 15  ................
000010  18 17 20 19 22 21 24 23 26 25 28 27 30 29 32 31  .. ."!$#&%('0)21
```

## Performing an Exclusive-OR Checksum

Use the Checksum (EXOR) command to calculate an exclusive-OR checksum of RAM. You may wish to use this command instead of the sumcheck (total) command if data being downloaded to the programmer includes an exclusive-OR checksum, or if you prefer to work with an exclusive-OR summation. The exclusive-OR checksum is a 1-byte hexadecimal value representing the result of exclusive-ORing each successive byte of the data range specified. For example,

```
  0000 0001
+ 1111 1111
  1111 1110
```

Command Format: CX <start address> <end address> [<destination address>]

| Procedure | Example Key Sequence |
|---|---|
| 1. Type in the Checksum (EXOR) command letters, CX. | CX |
| 2. Type in the address of the first byte of RAM to be checked by the checksum operation. | CX 0 |
| 3. Type a space and then type in the address of the last byte of RAM to be checked by the checksum operation. | CX 0 3FFF |

| Procedure | Example Key Sequence |
|---|---|
| 4. To perform the sumcheck without storing the sumcheck value in RAM, press carriage return, or, to store the sumcheck value in RAM, type a space, type in the RAM address that you want the sumcheck value to be stored in and press carriage return. If you store the sumcheck value in RAM, it will overwrite any data previously stored in the RAM locations to which the sumcheck is written. Make sure that there is no data that you want to retain located in the addresses that the sumcheck will be written to.<br><br>The sumcheck (total) is calculated and then displayed on the screen. The sumcheck is also stored in memory if you specified a destination address. It should be noted that while 4 bytes are calculated and stored by this command, only the first 3 bytes are displayed. | CT 0 3FFF<CR><br>or<br>CT 0 3FFF 4000<CR> |

## Example

Calculate the sumcheck total of memory addresses 0 through 3FFF and store the sumcheck value in address 4000:

```
>CT 0 3FFF 4000<CR>
THE SUMCHECK (TOTAL) IS:  030201
```

Display the sumcheck with an Intel data translation format selected:

```
>M 4000 4003<CR>
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
004000  01 02 03 04
```

*NOTE*

*The four-byte sumcheck is stored in memory addresses 4000 and 4003. If an Intel translation format is currently selected, the low-byte (01 in example above) is stored in 4000 and the high-byte (04 in example above) is stored in 4003. If a Motorola format is currently selected, the low-byte (01 in example) is stored in 4003 and the high-byte (04 in example) is stored in 4000.*

# 4. COMPUTER REMOTE CONTROL

## INTRODUCTION

The 288A Multi Programmer can be controlled via a remote host computer using the Computer Remote Control (CRC) commands described in this section. The Computer Remote Control commands were designed to be incorporated into a software program, or driver, which would allow an operator to control the 288A Multi Programmer using the software program. The driver generates and sends commands to the programmer which executes the commands and returns a response character, and in some cases also data, which the driver then reacts to and uses to generate messages and prompts for the operator.

The CRC, or driver, commands are ASCII characters which are summarized in the command summary table following this introduction and described in more detail in the pages that follow. The commands are grouped into the following subsections:

- ENTERING AND EXITING COMPUTER REMOTE CONTROL MODE—This subsection explains how to transfer control of the programmer to a remote host computer and how to return control of the programmer to the front panel.
- VERIFYING PROPER COMMUNICATION—This subsection describes the commands used to verify that proper communication has been established between the computer and the programmer.
- PROGRAMMING OPERATIONS—This subsection describes the commands used to set parameters prior to performing programming operations, load data into RAM, program blank devices, test devices and verify that devices were programmed properly.
- TRANSFERRING DATA—This subsection describes the commands used to upload or download data to or from the remote host computer.
- INQUIRING ABOUT OPERATING AND ERROR STATUS—This subsection describes the commands used to inquire about parameters and options selected and error status.
- DATA TRANSLATION FORMATS—This subsection describes the eight data translation formats available for transferring data to and from the 288A Multi Programmer through the RS-232C port.

| Procedure | Example Key Sequence |
|---|---|
| 4. To perform the checksum operation without storing the checksum value in RAM, press carriage return. To store the checksum in RAM, type a space, type the RAM address that you want the checksum to be stored in and press carriage return. If you store the checksum value in RAM, it will overwrite any data previously stored in that RAM location.<br><br>The checksum (EXOR) value is calculated and then displayed on the screen. The checksum is also stored in RAM if you specified a destination address. | CX 0 3FFF<CR><br>or<br>CX 0 3FFF 4000<CR> |

## Example

Calculate the checksum of memory addresses 0 through 3FFF and store the checksum value in address 4000:

```
>CX 0 3FFF 4000<CR>
THE CHECKSUM (EXOR) IS:  FE
```

## Computer Remote Control Command Summary

| Command Format | Description |
|---|---|
| M | Enter CRC mode (automatic setup of communicatons protocol) |
| Z | Exit CRC mode |
| <CR> | Execute command |
| <ESC> | Abort operation |
| hhU | Set nulls |
| H | No operation |
| ffpp@ | Select family and pinout codes |
| hhhhhh: | Set beginning device address |
| hhhhhh< | Set beginning RAM address |
| hhhhhh; | Set block size |
| nn22] | Convert to 16-bit or 32-bit mode |
| L | Load master device(s) |
| nP | Program a set of n devices |
| hh^ | Clear RAM with data "hh" |
| nT | Test n devices for illegal bits |
| B | Blank check all devices |
| nS | Sumcheck data of device in socket n or RAM (if n=0) |

| Command Format | Description |
|---|---|
| n23] | Set verify pass number (1 or 2) |
| hhll18] | Set Vcc level high (hh) and low (ll) |
| nV | Verify a set of n devices |
| n= | Enable/disable I/O timeout |
| hhhhhhhhW | Set I/O address offset |
| cnnA | Select control code (c) and data translation format (nn) |
| I | Input data from host computer |
| O | Output data to host computer |
| C | Compare data transferred |
| [ | View current family and pinout codes |
| CC] | View current electronic ID family and pinout codes |
| CD] | View current electronic ID code |
| R | Respond with device parameters |
| / | View number of devices failed |
| X | View last 16 error codes |
| G | View firmware configuration |

## Symbols and Conventions

The following is a list of symbols and conventions used in this section of the manual.

| | |
|---|---|
| A | Capital letters in a command must be sent to the programmer through the serial port in order to execute the command. |
| h | A lower-case h represents a hexadecimal data character. |
| n | A lower-case n represents a decimal digit. |
| ff | Other lower case letters, besides "h" and "n," represent other kinds of data, such as the family code (ff) or the pinout code (pp) of a device. |
| <CR> | This symbol represents a carriage return, which must follow each command entry. |
| <ESC> | This symbol represents the escape key. |

## Specifying Block Parameters

The block parameters "beginning RAM address," "beginning device address," "block size," and "I/O address offset" are defined in the following paragraphs. These definitions should help you determine how to select the correct parameter values for the operation being performed. Several examples are also given following the definitions which should show you how these parameters interrelate.

*NOTE*

*All addresses specified in the following paragraphs and examples are in hexadecimal notation.*

BEGINNING RAM ADDRESS—The beginning RAM address is the memory address from which the first data byte of a block of data is read during a transfer from RAM, or into which the first data byte of a block is written during a transfer to RAM. Data bytes are then transferred to or from subsequent RAM addresses in sequential order. In other words, if a beginning RAM address of 1000 is specified for a "Load Master Device" operation, the first data byte transferred from the master device into RAM will be written into RAM address 1000 (regardless of the device address that the data was read from). The second byte transferred to RAM will be written into address 1001, the next byte into RAM address 1002, and so on.

BEGINNING DEVICE ADDRESS—The beginning device address is the device's memory address from which the first data byte is read during a "Load from Device" operation, or into which the first byte of a block of data is written during a "Program" operation. Subsequent data bytes are then transferred to or from the device in sequential order. In other words, if a beginning device address of 00FF is specified for a "Load from Device" operation, the first byte of data transferred to programmer RAM would be read from device address 00FF. The second byte of data trasferred to RAM would be read from device address 0100, the next byte from device address 0101, and so on.

BLOCK SIZE—The block size is the number of bytes of data read from programmer RAM or written to programmer RAM, depending upon the operation being performed. For instance, if you specify a block size of 1000 (hexadecimal) and a beginning RAM address of 2000 for a "Load from Device" operation, 1000 bytes will be loaded from the the device into RAM addresses 2000 through 2FFF.

I/O ADDRESS OFFSET—The I/O address offset specifies the address which will be assigned to the first byte of programmer RAM data output to the host computer for an output (or upload) operation. The data in programmer RAM is translated into the specified data translation format prior to being output through the RS-232C port. The data is formatted into data records which include a "record address." The record

## Response Characters

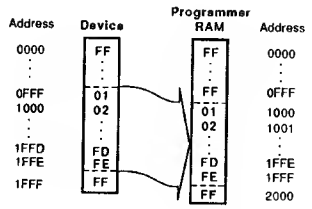
The response characters, summarized in the table below, are characters that the programmer sends to the host computer after attempting to execute a command. The programmer's response to a command will always contain a response character followed by a carriage return. In addition, the response may contain data and a line feed and nulls (ASCII character 00). Whether or not the response contains a line feed or nulls is dependent upon the null count setting. (How to set the null count is described in the Verifying Proper Communication subsection.) If you want to insure that you always get a line feed after you enter a CRC command, set the null count between "00" and "FE." The default null count, "FF," will not send a line feed after a command is entered.

| Character | Name | Description |
|---|---|---|
| > | Prompt | The programmer returns a prompt character when it enters Computer Remote Control mode, when <ESC> halts a command, or when the programmer successfully executes a command. The programmer then transmits a carriage return. |
| F | Fail | The programmer returns an "F" when it fails to execute a command. The programmer then transmits a carriage return. To find out what errors were produced by the command, use the "X" command to output the last 16 error codes produced. |
| ? | Question | The programmer returns a question mark when it does not understand a command or the command (as entered) was invalid. The programmer then transmits a carriage return. |

## Examples

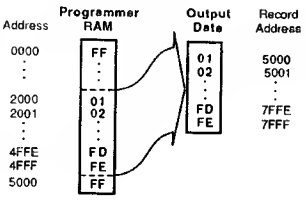
**Load Operation:**

Beginning RAM address = 1000
Beginning device address = 0FFF
Block size = 1000



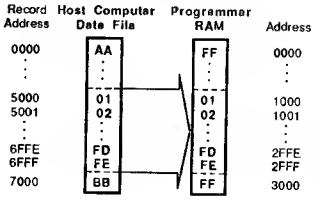

**Upload Operation:**

Beginning RAM address = 2000
Block size = 3000
I/O address offset = 5000




**Download Operation:**

Beginning RAM address = 1000
Block size = 2000
I/O offset address = 5000



address is the address assigned to the first byte of data contained in that record. The offset address specifies the first record address of the block of data being output (uploaded). For example, if you specify a beginning RAM address of 4000 and an offset address of 8000, the data from programmer RAM address 4000 will be assigned a record address of 8000. Consecutive RAM data bytes following the first data byte will be assigned consecutive record addresses (8001, 8002, 8003, and so on). The default output offset address is zero.

For input, the I/O address offset specifies the first data record of the data file in the host computer to be input (or downloaded). Each data record in the host computer data file contains a "record address" (which specifies the address of the first data byte in the record). The record address is assigned when the program is compiled into object code (or machine code) and then formatted into a data translation format. During a download operation, the offset address specifies the record address of the first data byte that will be stored in RAM. For example, if you specify an offset address of 3000 and a beginning RAM address of 4000, the data bytes contained in the record with record address 3000 (the offset address) will be stored in RAM starting at the beginning RAM address (4000). If the next data record has an assigned record address of 3010, the data bytes of that record will be stored in programmer RAM starting at RAM address 4010. (To determine the programmer RAM address into which the the first byte of data in a record will be stored, the offset address is subtracted from the record address and the result is added to the beginning RAM address.) Data bytes contained in records with subsequent addresses are stored in subsequent RAM addresses, up through the block size specified.

For input, the default offset address is "FFFFFFFF." This number has special meaning to the translators, acting as a flag and resulting in the I/O address offset being assigned to the value of the first incoming record address. This will cause the first byte of data received by the programmer to be stored in the beginning RAM address selected and subsequent data bytes received to be stored in subsequent RAM addresses. Data records with record addresses which are less than the offset address will be discarded by the programmer (not stored in RAM).

# ENTERING AND EXITING COMPUTER REMOTE CONTROL

The following procedures describe how to enable Computer Remote Control mode (transfer control of the programmer to the host computer) and return control of the programmer to the front panel.

## Entering Computer Remote Control

| Procedure | 288A Displays |
|---|---|
| 1. Connect the programmer to the computer as described in the RS-232C Port Cable Connections discussion in the Getting Started section of this manual. | |
| 2. Power up the programmer by pressing the power switch on the back of the programmer to the ON position. | SELF TESTING<br>. . . . . |
| 3. When the self test is complete and the display reads LOAD FROM MASTER, set the communications protocol of the programmer to match the communications protocol of the host computer (see the Front Panel Operation section for instructions on setting the 288A port settings), or skip to step 4 if the driver program sets the correct communications protocol automatically (using the "M<CR>" command). | LOAD FROM MASTER |
| 4. Scroll through the main menu to RS232 PORT and press ENTER. | RS232 PORT<br>COMPUTER CONTROL |

## Using PROMlink™ to Operate the 288A

If you are using PROMlink as the driver program to operate the 288A, you must set the programmer's serial port (I/O) settings to match PROMlink's I/O settings before entering CRC mode. Refer to the Front Panel Operation section for instructions on how to set the 288A's serial port (I/O) settings. In order to use the 288A with PROMlink, you must have at least version 2.0 of PROMlink. If your version of PROMlink does not support the 288A Multi Programmer specifically, you can still use PROMlink (version 2.0 or greater) to operate the 288A by selecting either the 288A Set Programmer or the Model 120/121 as the programmer type.

## Aborting an Operation

Any of the Computer Remote Control commands can be aborted by pressing <ESC>.

## VERIFYING PROPER COMMUNICATION

The first thing that the driver program should do is verify that the computer and the programmer are communicating properly. To do this, use the following commands.

| Command Format | Command Name | Data Output to Host | Description |
|---|---|---|---|
| hhU | Set Nulls | none | Sets the number of nulls (hh) sent to the host computer after the response character and carriage return and the number of nulls sent between data records upon output, and also enables line feeds. Values 0 through FE enable line feeds and cause the specified number of nulls (0 through FE) to be sent to the computer. The default value, FF, causes no nulls and no line feeds to be sent. |
| H | No Operation | none | Causes the programmer to return a prompt (>), indicating that communications are established. The driver program should send this command to the programmer upon entry of CRC mode to verify that communications are established. |

| Procedure | 288A Displays |
|---|---|
| 5. Scroll through the RS232 PORT menu to COMPUTER CONTROL and press ENTER. | COMPUTER CONTROL |
| 6. See the driver instructions for entering the driver program if you are using an existing driver.<br><br>If you are writing your own driver, you can establish communications between the programmer and the host computer by either of two methods: 1) You can cause the programmer to set the communications protocol automatically (so the operator does not have to set the communications protocol manually) by sending an "M<CR>"to the programmer, or 2) if the communications protocol will always beset correctly prior to this step, you can send any character to the programmer to establish communications. A prompt (>) character will be returned to the host computer if the port settings were set correctly and the programmer and computer are communicating.<br><br>The programmer will display COMPUTER CONTROL until control of the pmgrammer is returned to the front panel. | COMPUTER CONTROL |

## Exiting Computer Remote Control

To exit Computer Remote Control mode use the appropriate exit command specified by the CRC driver program, or press any programmer key. If you are writing your own driver, the driver's exit command must send a "Z<CR>" to the programmer to cause it to exit CRC mode. The user should still be able to exit CRC mode by pressing a programmer key when using any driver.

After exiting CRC mode, the programmer displays: RS232 PORT. You can now continue to operate the programmer using the front panel keys.

| Command Format | Command Name | Data Output to Host | Description |
|---|---|---|---|
| nn22J | Convert to 16-bit or 32-bit Mode | none | Causes the 288A to load, program and verify each pair of 8-bit-wide devices using a 16-bit-wide word format when nn=16; or, causes the 288A to load, program, or verify each set of four 8-bit-wide devices using a 32-bit-wide word format when nn=32. To return the programmer to 8-bit-wide mode, set nn equal to 08. See the Terminal Remote Control section for an explanation of device to memory mapping when using 16-bit-wide (word-wide) format or 32-bit-wide (long-word-wide) format. |
| **Programming Devices** | | | |
| L | Load Master Device(s) | none | Loads data into RAM from a master device or set of masters. The 288A automatically detects the number of masters installed and loads the entire set. |
| nP | Program Device(s) | none | Programs and verifies a single device or gangs of identical devices if n=1, or programs and verifies sets of n devices if n is 2 through 8. If 16-bit or 32-bit mode is selected, n equals the number of sets (not the total number of devices). The program sequence in CRC mode does not include a blank check or an illegal bit test. The default set size is one. |
| hh^ | Clear RAM | none | Clears (fills) all of RAM with data hh. The default clear data is 00. |

# PROGRAMMING OPERATIONS

Use the following commands to set the family and pinout codes, set programming parameters and options, load device data into memory, clear memory, program devices, test devices and verify programmed devices.

| Command Format | Command Name | Data Output to Host | Description |
|---|---|---|---|
| **Setting Programming Parameters and Options** | | | |
| ffpp@ | Select Family and Pinout Codes | none | Selects the family (ff) and pinout (pp) codes for the devices to be loaded, programmed, tested or verified. The correct family and pinout codes must be selected prior to programming a device, otherwise damage to the device could result. |
| hhhhhh: | Set Beginning Device Address | none | Sets the first device address from which or to which data is to be transferred. The default beginning device address is 0. |
| hhhhhh< | Set Beginning RAM Address | none | Sets the first RAM address to be used for data transfers. The default beginning RAM address is 0. |
| hhhhhh; | Set Block Size | none | Sets the block size (number of bytes) to be used in data transfers. The default block size is the device size times the set size (number of unique devices in the set). To reset the block size to the default, enter all zeroes. |

| Command Format | Command Name | Data Output to Host | Description |
|---|---|---|---|
| **Verifying Devices** | | | |
| n23] | Set Verify Pass Number | none | Sets the number of verify passes (n) to be made, either 1 or 2. |
| hhll18] | Set Vcc Level High (hh) and Low (ll) | none | Sets the high-pass (hh) and low-pass (ll) verify Vcc levels in tenths of volts. The allowable high-pass range is 50 to 60 and the allowable low-pass range is 40 to 50. |
| nV | Verify Device(s) | none | Compares RAM data with programmed devices or sets of n devices. The default set size is one. If 16-bit mode or 32-bit mode is selected, n equals the number of sets (not the total number of devices). |

| Command Format | Command Name | Data Output to Host | Description |
|---|---|---|---|
| **Testing Devices** | | | |
| nT | Test Device(s) for Illegal Bits | none | Tests single or gangs of devices (if n=1) or sets of n devices for illegal bits (programmed bits that do not exist in RAM). The default set size is one (single or gang test). If 16-bit or 32-bit mode is selected, n equals the number of sets (not the total number of devices). If an illegal bit is found, the 288A returns an "F" to the computer and stores the appropriate error code in programmer RAM. To output the stored error code, use the "X" command. |
| B | Blank Check All Devices | none | Checks the installed devices for programmed bits. If a programmed bit is found, the 288A returns an "F" to the computer and stores the appropriate error code in programmer RAM. To output the stored error code to the host computer, use the "X" command. |
| nS | Sumcheck Device or RAM | hhhh | Calculates the two byte hexadecimal sumcheck of the data loaded from the device in socket n; if n=0, "S" calculates the sumcheck of RAM data from the beginning RAM address (or zero if no beginning RAM address is specified) to the block size selected. If no block size is selected, the 288A uses the word limit of the selected device(s) as the block size. |

# INQUIRING ABOUT OPERATING AND ERROR STATUS

Use the following commands to cause the programmer to output the status of the current device(s), the electronic ID settings, error status, and the programmer firmware configuration.

| Command Format | Command Name | Data Output to Host | Description |
|---|---|---|---|
| [ | View Current Family and Pinout Codes | ffpp | Outputs the family (ff) and pinout (pp) codes of the device currently selected. |
| CC] | View Current Electronic ID Family and Pinout Codes | ffpp | Ouputs the family (ff) and pinout (pp) codes of the last electronic ID used in an operation. |
| CD] | View Current Electronic ID Code | h1...h16 | Outputs the 16-byte electronic identifier code of the installed device. |
| R | Respond with Device Parameters | wl/ws/n | Outputs the status of the device selected by the current family and pinout codes.<br><br>wl = word limit (3 or 4 hex digits)<br>ws = word size (4 or 8 bits)<br>n = VOL or VOH (1=VOL and 0=VOH) |
| / | View Number of Devices Failed | xx-yy | Outputs the number of devices that failed (xx) and the number of devices in the sockets (yy). |

## TRANSFERRING DATA

Use the following commands to input data to the programmer from the host computer or output data from the programmer to the host computer.

| Command Format | Command Name | Data Output to Host | Description |
|---|---|---|---|
| n= | Enable/Disable I/O Timeout | none | Enables the I/O timeout if n is set to any value other than 0. Timeout is disabled when n=0, which is the default. |
| hhhhhhhhW | Set I/O Address Offset | none | Specifies the record address of the first data record to be stored in RAM during an input operation and specifies the record address assigned to the first outgoing data byte during an output operation. The default address offset is the first address received for input, and zero for output. To reset to the default offset for input, enter all Fs for the address offset. |
| cnnA | Select Control Code and Data Translation Format | none | Selects the instrument control code (c) and data translation format (nn) used for input and output of data. See the Data Translation Formats subsection at the end of the CRC section for details on the instrument control code and data translation formats. The default format is fntel Intellec 8/MDS, format 83, and the default control code is 0. |
| I | Input Data From Host | none | Instructs the programmer to accept and store formatted data from the host computer. |
| O | Output Data to Host | hhhhh... | Translates data into the selected format and outputs this data to the computer. |
| C | Compare Data Transferred | none | Compares data in RAM with the data received through the serial port from the host computer. |

# DATA TRANSLATION FORMATS

## Introduction

This subsection describes the data translation formats used and recognized by the 288A Multi Programmer. The 288A Multi Programmer is capable of interfacing with most RS-232C serial equipment employing one of the data translation formats listed below. The format of the code generated depends on the system you are using. Check your System Manual to see which format(s) your system supports. Sample printouts of each of the 288A's supported formats are given on the pages that follow.

Before the formatted data may be transmitted, you must specify to the 288A the format in which the data will be sent. You can do this using either the Computer Remote Control command "cnnA," described in the previous subsection, or the programmer's front panel keys and menus. The code number assigned to each of the data translation formats is listed on the following page. When selecting the data translation format in CRC mode with the "cnnA" command, replace "nn" with the listed code corresponding to the desired data translation format. The formats are listed in alphabetical order.

| Command Format | Command Name | Data Output to Host | Description |
|---|---|---|---|
| X | View Error Codes | n1...n16 | Outputs the last 16 error codes (n1 through n16) which occurred. This command also clears the error codes from memory. See the Error Messages section for explanations of the meanings of the error codes. |
| G | View Firmware Configuration | hhhhhh | Outputs the three-byte configuration number of the programmer's firmware. |

## Software Handshaking

In addition to the 2-digit data translation format code associated with each translator, a 1-digit control code may immediately precede the format code to signal or control the data transfer. The codes must be formatted as follows: "cnn," where "c" is the instrument control code and "nn," the format code. If no codes are entered into the programmer, the current default values will be used. The three possible values of the control code and their functions are shown in the following table.

| Control Code | Name | Input Function | Output Function |
|---|---|---|---|
| 0 | Handshake Off | Data is received without handshaking. | Data transmission will be halted upon receipt of an "X-OFF" character; transmission will resume upon receipt of an "X-ON" character. |
| 1 | Handshake On | Transmit an "X-ON" character when ready to receive data; transmit an "X-OFF" character if the receiver buffer is full; transmit an "X-ON" character if the receiver buffer is empty; transmit an "X-OFF" character after all of the data is received. | Transmit a "PUNCH ON" character prior to data transmission. Data transmission will be halted upon receipt of an "X-OFF" character; transmission will resume upon receipt of an "X-ON" character. A "PUNCH OFF" character is sent when the transmission is completed. |

| Format | Code |
|---|---|
| ASCII Space Hex | 50 |
| Binary | 10 |
| Intel 32-bit Hexadecimal | 99 |
| Intel Intellec 8/MDS | 83 |
| Intel MCS-86 Hexadecimal Object | 88 |
| Motorola Exorciser (S1) | 82 |
| Motorola Exormax (S1 and S2) | 87 |
| Motorola 32-bit (S1, S2 and S3) | 95 |
| Tektronix Hexadecimal | 86 |

## Hardware Handshaking

Hardware handshaking may be used if compatible with the host interface by connecting the Request to Send and Clear to Send lines at the serial interface ports. This type of handshaking does not require user selection, as does software (XON/XOFF) handshaking. To set up hardware handshake, simply connect the proper wires at each of the serial interface ports. See the Getting Started section for details on connecting the RS-232C port in either a hardware handshake or non-handshake mode.

## Leader/trailer and Null Output

A leader/trailer is a string of characters that is attached to the beginning and end of a data file. It is used to separate different files from one another and allows extra room which may be necessary for loading and unloading the data medium to or from equipment. For the 288A Multi Programmer, the leader is sent at the beginning and end of a data output operation. This leader will always be comprised of a carriage return, a line feed, and 50 nulls in succession, with one exception. If the null count parameter is set to "FF," the leader/trailer is skipped.

Null count is the number of null characters in the string of characters between each record or line within a file. What actually comprises a data record depends upon the format that is being used. Records and lines can basically be thought of as separations of data within a file. Parity for the beginning and end leader is the same as the parity for the data within the file. The same is true for the carriage return, line feed and nulls separating the records or lines of the file.

| Control Code | Name | Input Function | Output Function |
|---|---|---|---|
| 2 | X-ON/X-OFF | Data is received without handshaking. | Transmit data only after receiving an "X-ON" character. Data transmission will be halted upon receipt of an "X-OFF" character; transmission will resume upon receipt of an "X-ON" character. |

*NOTE*

*"X-ON" character is a <Ctrl-Q>, DC1 or 11 hex*
*"X-OFF" character is a <Ctrl-S>, DC3 or 13 hex*
*"PUNCH-ON" character is a <Ctrl-R>, DC2 or 12 hex*
*"PUNCH-OFF" character is a <Ctrl-T>, DC4 or 14 hex*

## ASCII Space Hex, Code 50

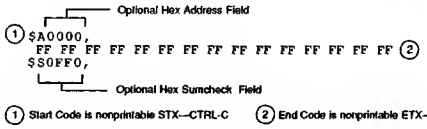
This format begins with a nonprintable STX (CTRL-B) start code and ends with a nonprintable ETX (CTRL-C) end code and can also contain an optional address field and sumcheck field. The illustration shows data bytes formatted in ASCII Space Hex. Data in this format is organized in sequential bytes separated by the execute character (space). Characters immediately preceding the execute character are interpreted as data. This format expresses 8-bit data by 2 hex characters. Line feeds, carriage returns and other characters may be included in the data stream as long as a data byte directly precedes each execute character.

Although each data byte has an address, most are implied. Data bytes are addressed sequentially unless an explicit address is included in the data stream. This address is preceded by a "$" and an "A," must contain 2 to 8 hex characters, and must be followed by a comma. The programmer skips to the new address to store the next data byte; succeeding bytes are again stored sequentially.

This format has an end code which terminates input operations. However, if a new start code follows within 16 characters of an end code, input will continue uninterrupted.

After receiving the final end code following an input operation, the programmer calculates a sumcheck of all incoming data. Optionally, a sumcheck can also be entered immediately following the end code. The sumcheck field consists of 2 to 4 hex characters, sandwiched between a "$" and a comma. The programmer compares this sumcheck with its own calculated sumcheck. If they match, the programmer will display the sumcheck, if not, a sumcheck error will be displayed. It is optional to include the sumcheck in data being input to the programmer, but a sumcheck is always included with data output from the programmer.

Output is begun by invoking an output, or upload, operation. The programmer divides the output data into 8-line blocks. Data transmission is begun with the start code, a nonprintable STX. Data blocks follow, each one prefaced by an address for the first data byte in the block. The end of transmission is signalled by the end code, a nonprintable ETX. Directly following the end code is a sumcheck of the transferred data.




(1) Start Code is nonprintable STX—CTRL-C (2) End Code is nonprintable ETX—CTRL-C

## Data Verification

For data verification, the 288A calculates a sumcheck of all data sent to or from the programmer. At the end of a successful input operation, the programmer will output the sumcheck of all data transferred. It will also compare any received sumcheck fields with its own calculation. If the two agree, the programmer will output the sumcheck; a mismatch will produce an error message. Output data is always followed by a sumcheck field which may be printed on disk or tape for use in subsequent input operations.

## Formats with Limited Address Fields

Some formats are not defined for use with address fields greater than 64K. Thus, if you transfer a block greater than 64K, the address fields that would be greater than 64K may wrap around and overwrite data transferred in previous data records. The formats that may exibit this characteristic are 82, 83 and 86.

## Intel Hex-32, Code 99

The Intel 32-bit Hexadecimal Object file record format has a 9-character (4-field) prefix that defines the start of record, byte count, load address, and record type and a 2-character checksum suffix. The figure illustrates the sample records of this format.

The six record types are:

00 = data record
01 = end record (signals end of file)
02 = extended segment address record (added to the offset to determine the absolute destination address)
03 = start segment address record (not sent during output by programmer)
04 = extended linear address record (added to the offset to determine the absolute destination address)
05 = start linear address record (not sent during output by programmer)

Record type 00, the data record, begins with the colon start character. This is followed by the byte count (in hex notation), the address of the first data byte, and the record type (equal to "00"). Following these are the data bytes. The checksum follows the data bytes and is the two's complement (in binary) of the preceding bytes in the record, including the byte count, address, record type and data bytes.

Record type 01, the end-of-file record, also begins with the colon start character. This is followed by the byte count (equal to "00"), the address (equal to "0000"), the record type (equal to "01") and the checksum, "FF".

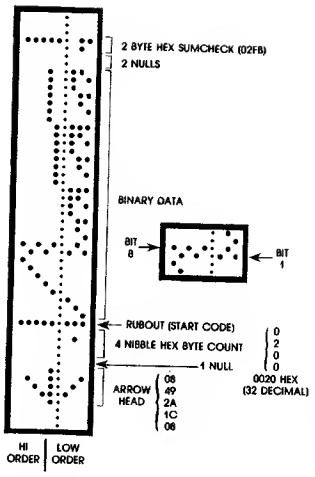
## Binary Transfer, Code 10

Data transfer in the Binary format consists of a stream of 8-bit data words preceded by a byte count and followed by a sumcheck. The Binary format does not have addresses.

A paper tape generated by a programmer will contain a 5-byte, arrow-shaped header followed by a null and a 4-nibble byte count. The start code, an 8-bit rubout, follows the byte count. The end of data is signalled by 2 nulls and a 2-byte sumcheck of the data field. Refer to the illustration.

The programmer stores incoming binary data upon receipt of the start character. Data is stored in RAM starting at the first RAM address and ending at the last incoming data byte.

The standard Binary Transfer format (with the arrow header, byte count and sumcheck) can only be used for data in block sizes of 64K or less because a larger byte count could not fit into the 4-nibble byte count field.



### Example

Problem: Find the address for the first data byte for the following file.

```
:02 0000 04 0010 EA
:02 0000 02 1230 BA
:10 0045 00 55AA FF ..... BC
```

Solution:

Step 1: Find the extended linear address offset for the data record (1010 in the example).

Step 2: Find the extended segment address offset for the data record (1230 in the example).

Step 3: Find the address offset for the data from the data record (0045 in the example).

Step 4: Calculate the absolute address for the first byte of the data record as follows:

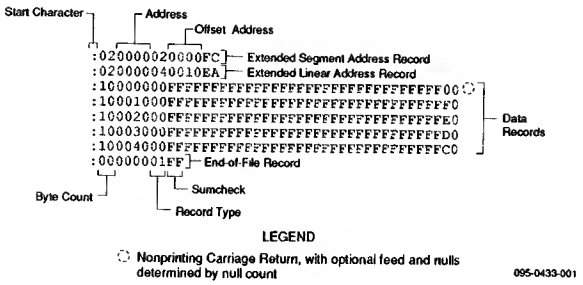
| | | |
|---|---|---|
| | 00100000 | linear address offset, shifted left 16 bits |
| + | 12300 | segment address offset, shifted left 4 bits |
| + | 0045 | address offset from data record |
| = | 00112345 | 32-bit address for first data byte |

The address for the first data byte is therefore 112345.

*NOTE*

*Always specify the address offset when using this format, even when the offset is zero.*

During output translation, the firmware will force the record size to 16 (decimal) if the record size is specified greater than 16. There is no such limitation for record sizes specified less than 16.




Record type 02, the extended segment address record, defines bits 4 to 19 of the segment base address. It can appear randomly anywhere within the object file and affects the absolute memory address of subsequent data records in the file. The address field for this record must contain ASCII zeros (Hex 30's). The following example illustrates how the extended segment address is used to determine a byte address.

Record type 03, the start segment address record, specifies bits 4-19 of the execution start address for the object file. This record is not used by programmer.

Record type 04, the extended linear address record, specifies bits 16-31 of the destination address for the data records that follow. It can appear randomly anywhere within the object file. The address field for this record must contain ASCII zeros (Hex 30's).

Record type 05, the start linear address record, specifies bits 16-31 of the execution start address for the object file. This record is not used by programmer.

## Intel MCS-86 Hexadecimal Object, Code 88

The Intel 16-bit Hexadecimal Object file record format has a 9-character (4-field) prefix that defines the start of record, byte count, record address, and record type and a 2-character hexadecimal sumcheck suffix. The illustration shows a valid data file of this format.

The four record types are:

00 = data record
01 = end record (signals end of file)
02 = extended address record (added to the offset to determine the absolute destination address)
03 = start record (ignored during input and not sent during output by Data I/O translator firmware)

Record type 00, the data record, begins with the colon start character. The colon is followed by the 2-character byte count (in hexadecimal notation), the 4-character address of the first data byte, and the record type, "00." This 9-character prefix is followed by the data bytes. The sumcheck follows the data bytes and is the binary two's complement of the preceding bytes in the record (including the byte count, address and data bytes) expressed in hexadecimal.

Record type 01, the end-of-file record, also begins with the colon start character. The colon is followed by the byte count, "00," the address, "0000," the record type, "01," and the sumcheck, "FF."

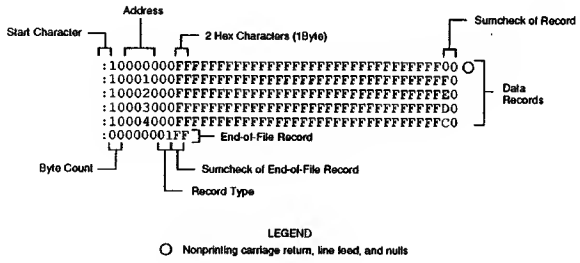
Record type 02, the extended address record, defines bits 4 to 19 of the record address. It can appear randomly anywhere within the data file and in any order; i.e., it can be defined such that the data bytes at high addresses are sent before the bytes at lower addresses. The example following the illustration shows how the extended address is used to determine a byte address.

## Intel Intellec 8/MDS Format, Code 83

Intel data records begin with a 9-character prefix, followed by data bytes, and end with a 2-character suffix. The illustration shows a valid Intel Intellec data file.

Each record begins with a colon, which is followed by a 2-character byte count. The 4 digits following the byte count give the address of the first data byte. The last 2 digits of the 9-character prefix are the record type. The record type is "00" for data records. Data bytes follow the 9-character prefix. Each data byte is represented by 2 hexadecimal digits ("FF" in the illustration); the number of data bytes in each record must equal the byte count. Following the data bytes of each record is the sumcheck (the binary two's complement of the preceding bytes, including the byte count, address and data bytes) expressed in hexadecimal.

The end-of-file record consists of the "colon" start character, the byte count "00," the address, the record type, "01," and the sumcheck of the record.



### Example

Problem: Find the address for the first data byte for the following file.

```
: 02 0000 02 1230 BA
: 10 0045 00 55AA FF .....BC
```

Solution:

Step 1: Find the record address for the byte. The first data byte is 55. Its record address is 0045 from above.

Step 2: Find the offset address. The offset address is 1230 from above.

Step 3: Shift the offset address one place left, then add it to the record address, like this:

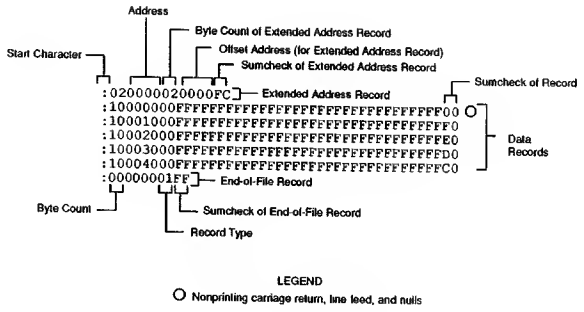
| | | |
|---|---|---|
| offset address | 1230 | (upper 16 bits) |
| + record address | 0045 | (lower 16 bits) |
| | 12345 | (20-bit address) |

The address for the first data byte is therefore 12345.

*NOTE*

*Always specify the address offset when using this format, even when the offset is zero.*

During output translation, the firmware will force the record size to 16 (decimal) if the record size is specified greater than 16. There is no such limitation for record sizes specified less than 16.

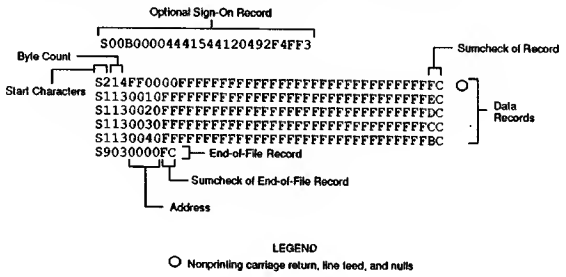

Address
Byte Count of Extended Address Record
Start Character
Offset Address (for Extended Address Record)
Sumcheck of Extended Address Record
Sumcheck of Record
:020000020000FC
Extended Address Record
:10000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00
:10001000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0
:10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
:10004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
Data Records
:00000001FF
End-of-File Record
Byte Count
Sumcheck of End-of-File Record
Record Type
LEGEND
Nonprinting carriage return, line feed, and nulls

## Motorola Exormax (S1 and S2) Format, Code 87

Motorola data files may begin with an optional sign-on record, initiated by the 2 start characters "S0." Data records start with an 8- or 10-character prefix and end with a 2-character suffix. The illustration shows a series of valid Motorola Exormax data records.

Each data record begins with 2 start characters, "S1" or "S2;" "S1" if the address field has 4 characters, "S2" if it has 6 characters. The third and fourth characters of the prefix represent the byte count, which expresses the number of data bytes plus 3, if the start characters were "S1," or plus 4, if the start characters were "S2." The address of the first data byte in the record is expressed by the last 4 characters of the prefix, if the address is "FFFF" or below, or the last 6 characters of the prefix, if the addresses is above "FFFF" (64K). Data bytes follow the prefix; each data byte is represented by 2 hexadecimal characters ("FF" in the illustration). The suffix is a 2-character sumcheck, which equals the binary one's complement of the summation of the preceding bytes in the record (including the byte count, address and data bytes) expressed in hexadecimal.

The end-of-file record begins with either the start characters "S8" or "S9." The start characters must be "S9" if the previous data record started with an "S1;" otherwise, either "S8" or "S9" may be used. Following the start characters are the byte count, "03," the address, "0000," and a sumcheck.

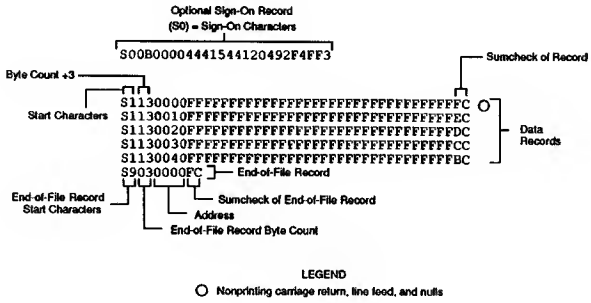


## Motorola Exorciser (S1) Format, Code 82

Motorola Exorciser data files may begin with an optional sign-on record, which is initiated by the start characters "S0." Valid data records start with an 8-character prefix and end with a 2-character suffix. The illustration shows a series of valid Motorola data records.

Each data record begins with the 2 start characters "S1." The third and fourth characters represent the byte count, which expresses the number of data bytes in the record (in hexadecimal) plus 3. The last 4 characters of the prefix represent the address of the first data byte in the record . Data bytes follow the address; each data byte is represented by 2 hexadecimal characters ("FF" in the illustration). The suffix is a 2-character sumcheck, which equals the binary one's complement of the summation of the byte count, address and data bytes, expressed in hexadecimal.

The end-of-file record consists of the 2 start characters "S9," the byte count, "03," the address (in hexadecimal) and a sumcheck.

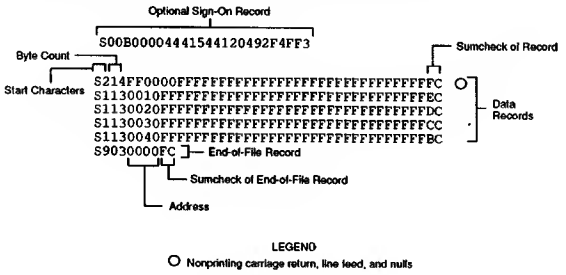


## Motorola Exormax (S1 and S2) Format, Code 87

Motorola data files may begin with an optional sign-on record, initiated by the 2 start characters "S0." Data records start with an 8- or 10-character prefix and end with a 2-character suffix. The illustration shows a series of valid Motorola Exormax data records.

Each data record begins with 2 start characters, "S1" or "S2;" "S1" if the address field has 4 characters, "S2" if it has 6 characters. The third and fourth characters of the prefix represent the byte count, which expresses the number of data bytes plus 3, if the start characters were "S1," or plus 4, if the start characters were "S2." The address of the first data byte in the record is expressed by the last 4 characters of the prefix, if the address is "FFFF" or below, or the last 6 characters of the prefix, if the addresses is above "FFFF" (64K). Data bytes follow the prefix; each data byte is represented by 2 hexadecimal characters ("FF" in the illustration). The suffix is a 2-character sumcheck, which equals the binary one's complement of the summation of the preceding bytes in the record (including the byte count, address and data bytes) expressed in hexadecimal.

The end-of-file record begins with either the start characters "S8" or "S9." The start characters must be "S9" if the previous data record started with an "S1;" otherwise, either "S8" or "S9" may be used. Following the start characters are the byte count, "03," the address, "0000," and a sumcheck.



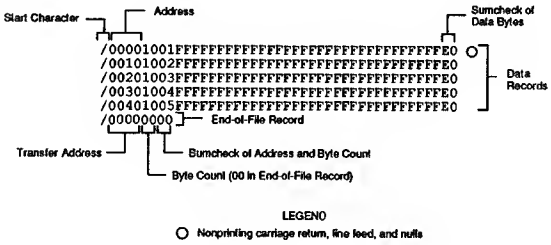
## Tektronix Hexadecimal Format, Code 86

Tektronix Hexadecimal records begin with a 9-character (4-field) prefix followed by data and end with a 2-character suffix. The figure illustrates a valid Tektronix data file.

Each data record begins with a slash start character. The next 4 characters of the prefix following the start character express the address of the first data byte. The address is followed by a 2-character byte count, which represents the number of data bytes in the record, and by a 2-character sumcheck of the address and byte count. Data bytes follow the sumcheck, represented by pairs of hexadecimal characters ("FF" in the illustration). Succeeding the data bytes is their sumcheck, an 8-bit sum, modulo 256, of the 4-bit hexadecimal values of the digits making up the data bytes. The sumcheck is expressed as a 2-character hexadecimal number. All records are followed by a carriage return.

The end-of-file record consists of a start character (slash), followed by the transfer address, the byte count, "00," and the sumcheck of the transfer address and byte count.

An optional abort record contains 2 start characters (slashes), followed by an arbitrary string of ASCII characters.

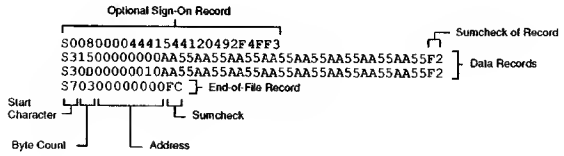


## Motorola 32-Bit (S3) Format, Code 95

The Motorola 32-bit (S3) format closely resembles the Motorola Exormax format — the main difference being the addition of the "S3" and "S7" start characters. Motorola data files may begin with an optional sign-on record, initiated by the start characters "S0" (zero). "S1" and "S2" record types are allowed using this format (see the Motorola Exormax Format section for an explanation of these record types). The following paragraphs describe the "S3" format. A sample transmission of the "S3" format is shown below.

The "S3" character is used to begin a data record containing a 4-byte address. The third and fourth characters of the record represent the byte count (which expresses the number of data bytes plus 5), followed by the address and sumcheck bytes in the record. The address of the first data byte in the record is expressed by the last 8 characters of the prefix. Data bytes follow the prefix; each data byte is represented by 2 hexadecimal characters. The number of data bytes occurring must be 5 less than the byte count. The suffix is a 2-character sumcheck — the one's complement (in binary) of the preceding bytes in the record, including the byte count, address and data bytes.

The "S7" character starts a termination record for a block of "S3" records. The address field for an "S7" record may optionally contain the 4-byte instruction address that identifies where control is to be passed.



# 5. ERROR MESSAGES

The following is a list of error codes and their corresponding messages. The circumstances which cause an error message to be displayed are described in the "Description" column and corrective action to take upon receiving the error message is explained in the right-hand column. The error codes are listed in numerical order.

*NOTE*

*Many of the pin driver errors are specific to the installed module. If you receive an error code and message that do not match any errors listed here, check your supplementary documentation for additional error code listings.*

| Code | Name | Description | Corrective Action |
|---|---|---|---|
| 15 | DRAM ERROR | The self test shows a dynamic RAM read/write verification error. | Contact your local Data I/O Service Center. |
| 17 | BANK ERROR | The bank selector of the DRAM circuit is unable to switch banks. | Contact your local Data I/O Service Center. |
| 18 | VCC ERROR | When 5 Vcc is applied to pin 26/28, the voltage drops by an unacceptable amount. | Indicates that the device is faulty. Replace the device. |
| 19 | VPP ERROR | The programming power supplies cannot be set at the proper levels. All of the socket LEDs light red. | Indicates that the device is faulty. This error cannot be isolated to a single socket, so if more than one device is installed, check each device individually to determine which device(s) are faulty. Replace the faulty device(s). |

| Code | Name | Description | Corrective Action |
|---|---|---|---|
| 29 | VERIFY ERROR | The programmed device data failed to verify against the master data in RAM. | Reprogram the device if possible or try programming another device. |
| 31 | NOT TRISTATE | The device failed to tri-state all data pins. | Indicates that the device or the device insertion is faulty. Discard the device or re-insert the device and attempt the operation again. |
| 32 | DEVICE NOT ENABLED | A device operation was performed and no device was installed, or not all of the device's data pins are driving when the device is enabled. | Install a device or devices and perform the operation again, or replace the faulty device. |
| 33 | DATA SHORT | The device contains a short on its data line. | Indicates that the device is faulty. Replace the device. |
| 34 | ADDRESS SHORT | The device contains a short on its address line. All of the socket LEDs light red. | Indicates that the device is faulty. This error cannot be isolated to a single socket, so if more than one device is installed, check each device individually (make sure that each device is installed correctly) and replace the faulty device(s). |

| Code | Name | Description | Corrective Action |
|---|---|---|---|
| 20 | NONBLANK DEVICE | Device failed blank test. | Press ENTER to repeat the blank check with the same device type selected. To exit the blank check operation, press a scroll key or a hexadecimal keypad key. |
| 21 | ILLEGAL BIT | Unable to program device due to already programmed bit(s) of incorrect polarity. | Erase the device if possible or discard it. |
| 22 | EEPROM FAILURE | EEPROM device failed to program properly. | Indicates that the device is faulty. Replace the device. |
| 25 | BAD DEVICE CODE | An invalid family and pinout code combination was entered. | Consult the Device List for valid family and pinout codes and enter the correct code for the installed device(s). |
| 26 | WRONG SOCKET | A load operation from a single master was specified, but the device was not in socket 1. | Remove the device and re-install it in socket 1. |
| 27 | BLOCK LIMIT ERROR | Programmer RAM size is insufficient to perform the current operation using the begin RAM address, block size, or set size specified. | Check the begin RAM address, block size, and set size to be sure they are set correctly. |

| Code | Name | Description | Corrective Action |
|---|---|---|---|
| 46 | I/O TIMEOUT | No characters, or only nulls and rubouts, were received upon serial input for 25 seconds after pressing the ENTER key; or, no characters could be transmitted for a period of 25 seconds due to the state of the handshake lines. | Check all connections and attempt transmission again. |
| 51 | I/O FORMAT ERROR | The programmer received an invalid address field. | Check all connections, check the data format and data source and then attempt transmission again. |
| 52 | I/O VERIFY FAILURE | The data from the serial port does not match the data in RAM. | Reload data into RAM. If the problem persists, service the programmer or contact your localData I/O Service Center. |
| 55 | I/O FORMAT ERROR | The sumcheck field received by the programmer does not agree with its own calculated sumcheck. | Reload data into RAM. If the problem persists, service the programmer or contact your local Data I/O Service Center. |
| 57 | I/O FORMAT ERROR | Data sent to the programmer are non-hexadecimal characters. Input data is in the incorrect format. | Specify the correct data translation format for the data being transferred, or correct the data in the file. |
| 5C | PIN 2 ERROR | The self test indicates that the socket pin 2 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |

| Code | Name | Description | Corrective Action |
|---|---|---|---|
| 35 | DATA BUS FAIL | Indicates that the device data bus may be damaged or the contact to the ZIF socket is faulty. | Make sure that the device(s) are installed in the sockets correctly, replace the device if faulty, or contact your local Data I/O Service Center. |
| 36 | BAD INSERTION | The device was inserted in the socket incorrectly. | Align the bottom-most pin of the device with the bottom of the socket. Pin 1 should be at the top of the device. |
| 38 | OVERCURRENT | The device to be programmed is drawing excessive current. | Indicates that the device is faulty. Replace the device. |
| 41 | FRAME ERROR | The serial interface detected a start bit but the stop bit was in the wrong position. | Check the current baud rate and stop bit settings and attempt transmission again. |
| 42 | OVERRUN ERROR | The serial interface received characters when the programmer was unable to service them. | Check the serial port connections. Make sure handshake lines are properly connected and attempt transmission again. |
| 43 | PARITY ERROR | The incoming data has incorrect parity. | Check the parity setting and attempt transmission again. |

| Code | Name | Description | Corrective Action |
|---|---|---|---|
| 68 | PIN 35 ERROR | The self test indicates that the socket pin 35 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 70 | PIN 22 ERROR | The self test indicates that the socket pin 22 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 71 | PIN 1VPP ERROR | The self test indicates that the 1VPP pin driver and control signals are at an inaccurate level. | Contact your local Data I/O Service Center. |
| 72 | PIN 1 ERROR | The self test indicates that the socket pin 1 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 74 | PIN 23 ERROR | The self test indicates that the socket pin 23 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 75 | PIN 24 ERROR | The self test indicates that the socket pin 24 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 76 | PIN 25 ERROR | The self test indicates that the socket pin 25 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 77 | PIN 28 ERROR | The self test indicates that the socket pin 28 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |

| Code | Name | Description | Corrective Action |
|---|---|---|---|
| 5F | PIN 26 ERROR | The self test indicates that the socket pin 26 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 60 | PIN 27 ERROR | The self test indicates that the socket pin 27 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 62 | PIN 29 ERROR | The self test indicates that the socket pin 29 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 63 | PIN 20 ERROR | The self test indicates that the socket pin 20 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 64 | PIN 31 ERROR | The self test indicates that the socket pin 31 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 65 | PIN 32 ERROR | The self test indicates that the socket pin 32 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 66 | PIN 33 ERROR | The self test indicates that the socket pin 33 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 67 | PIN 34 ERROR | The self test indicates that the socket pin 34 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |

| Code | Name | Description | Corrective Action |
|---|---|---|---|
| 90 | VCC FAIL | The Vcc supply circuits are not working properly. | Contact your local Data I/O Service Center. |
| 91 | DAC FAIL | The digital-to-analog converter circuit failed. | Contact your local Data I/O Service Center. |
| 92 | 5.6 V FAIL | The 5.6 V power supply circuit failed. | Contact your local Data I/O Service Center. |
| 94 | VPP FAIL | The programming pulse power supply circuit failed | Contact your local Data I/O Service Center. |
| A3 | PIN 36 ERROR | The self test indicates that the socket pin 36 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| A4 | PIN 37 ERROR | The self test indicates that the socket pin 37 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| A5 | PIN 38 ERROR | The self test indicates that the socket pin 38 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| A6 | PIN 39 ERROR | The self test indicates that the socket pin 39 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| A7 | PIN 40 ERROR | The self test indicates that the socket pin 40 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |

| Code | Name | Description | Corrective Action |
|---|---|---|---|
| 7A | PIN 21 ERROR | The self test indicates that the socket pin 21 driver is at an inaccurate level. | Contact your local Data I/O Service Center. |
| 81 | NO ID FOUND | The installed device has no electronic ID. | Check the Device List to make sure that the installed device supports the electronic ID feature. If the device does not have an electronic ID, use the correct family and pinout codes listed in the Device List. |
| 82 | INVALID ID | The electronic ID family and pinout codes of each of the devices installed do not match, or the device is not supported by this version of firmware. This error will only occur when the electronic ID mode is selected. | Consult the Device List for the correct family and pinout codes for all of the devices installed and remove any devices with incompatible family and pinout codes, or select the correct family and pinout codes of the device without using the electronic ID. |
| 83 | 8253 FAIL | Indicates a failure of an internal 288A 8253 device. | Contact your local Data I/O Service Center. |
| 84 | 8259 FAIL | Indicates a failure of an internal 288A controller device. | Contact your local Data I/O Service Center. |
| 89 | VPPA FAIL | The auxiliary VPP supply circuit is not working properly. | Contact your local Data I/O Service Center. |

## E



## F



## G



## H



## I

# INDEX

## L



## M



## N

## T